Equilibrium Computation and Robust Optimization in Zero Sum Games with Submodular Structure

Bryan Wilder

Department of Computer Science and Center for Artificial Intelligence in Society
University of Southern California
bwilder@usc.edu

Abstract

We define a class of zero-sum games with combinatorial structure, where the best response problem of one player is to maximize a submodular function. For example, this class includes security games played on networks, as well as the problem of robustly optimizing a submodular function over the worst case from a set of scenarios. The challenge in computing equilibria is that both players’ strategy spaces can be exponentially large. Accordingly, previous algorithms have worst-case exponential runtime and indeed fail to scale up on practical instances. We provide a pseudopolynomial-time algorithm which obtains a guaranteed \((1 - 1/e)^2\)-approximate mixed strategy for the maximizing player. Our algorithm only requires access to a weakened version of a best response oracle for the minimizing player which runs in polynomial time. Experimental results for network security games and a robust budget allocation problem confirm that our algorithm delivers near-optimal solutions and scales to much larger instances than was previously possible.

Introduction

Submodular functions are ubiquitous due to wide-spread applications ranging from machine learning, to viral marketing, to mechanism design. Intuitively, submodularity captures diminishing returns (formalized later). In this paper, we use techniques rooted in submodular optimization to solve previously intractable zero-sum games. We then show how to instantiate our algorithm for two specific games, including the robust optimization of a submodular objective.

As an example, consider the network security game introduced by Tsai et al. (2010). A defender can place checkpoints on \(k\) edges of a graph. An attacker aims to travel from a source node to any one of several targets without being intercepted. Each player has an exponential number of strategies since the defender may choose any set of \(k\) edges and the attacker may choose any path. Hence, previous approaches to computing the optimal defender strategy were heuristic with no approximation guarantee, or else provided guarantees but ran in worst-case exponential time (Jain et al. 2011; Iwashita et al. 2016).

However, this game has useful structure. The defender’s best response to any attacker mixed strategy is to select the edges which are most likely to intersect the attacker’s chosen path. Computing this set is a submodular optimization problem (Jain, Conitzer, and Tambe 2013). We give a general algorithm for computing approximate minimax equilibria in zero-sum games where the maximizing player’s best response problem is a monotone submodular function. Our algorithm obtains a \((1 - \frac{1}{e})^2\)-approximation (modulo an additive loss of \(\epsilon\)) to the maximizing player’s minimax strategy. This algorithm runs in pseudopolynomial time even when both action spaces are exponentially large given access to a weakened form of a best response oracle for the adversary. Pseudopolynomial means that the runtime bound depends polynomially on largest value of any single item (which we expect to be a constant for most cases of interest). Our algorithm approximately solves a non-convex, non-smooth continuous relaxation and then rounds the solution to a pure strategy in a randomized fashion. To our knowledge, no subexponential algorithm was previously known for this problem with exponentially large strategy spaces. Our framework has a wide range of applications, corresponding to the ubiquitous presence of submodular functions in artificial intelligence and algorithm design.

One prominent application is robust submodular optimization. A decision maker is faced with a set of submodular objectives \(f_1 \ldots f_m\). They do not know which objective is the true one, and so would like to find a decision maximizing \(\min_i f_i\). Robust submodular optimization has many applications because uncertainty is so often present in decision-making. We start by studying the randomized version of this problem, where the decision maker may select a distribution over actions such that the worst case expected performance is maximized (Krause, Roper, and Golovin 2011; Chen et al. 2017; Wilder et al. 2017). This is equivalent to computing the minimax equilibrium for a game where one player has a submodular best response. Our techniques for solving such games also yield an algorithm for the deterministic robust optimization problem, where the decision maker must commit to a single action. Specifically, we obtain bicriteria approximation guarantees analogous to previous work (Krause et al. 2008) under significantly more general conditions.

We make three contributions. First, we define the class of submodular best response (SBR) games, which includes the above examples. Second, we introduce the EQUATOR algo-
rithm to compute approximate equilibrium strategies for the maximizing player. Third, we give example applications of our framework to problems with no previously known approximation algorithms. We start out by showing that network security games (Tsai et al. 2010) can be approximately solved using EQUATOR. We then introduce and solve the robust version of a classical submodular optimization problem: robust maximization of a coverage function (which includes well-known applications such as budget allocation and sensor placement). Finally, we experimentally validate our approach for network security games and robust budget allocation. We find that EQUATOR produces near-optimal solutions for the attacker. We call an SBR game with $I = \{ S \subseteq X : |S| \leq k \}$ and $F = \{ f_P : P \text{ is a path from } s \text{ to } t \}$.

**Robust optimization setting:** One prominent application of SBR games is robust submodular optimization. Robust optimization models decision making under uncertainty by specifying that the objective is not known exactly. Instead, it lies within an uncertainty set $\mathcal{U}$ which represents the possibilities that are consistent with our prior information. Our aim is to perform well in the worst case over all objectives in $\mathcal{U}$. We can view this as a zero sum game, where the decision maker chooses a distribution over actions and nature adversarially chooses the true objective from $\mathcal{U}$. A great deal of recent work has been devoted to the setting of randomized actions, both because randomization can improve worst-case expected utility (Delage, Kuhn, and Wiesemann 2016), and
because the randomized version often has much better computational properties (Krause, Roper, and Golovin 2011; Orlin, Schulz, and Udwni 2016). Randomized decisions also naturally fit a problem setting where the decision maker will take several actions and wants to maximize their total reward. Any single action might perform badly in the worst case; drawing the actions from a distribution allows the decision maker to hedge their bets and perform better overall.

Previous work

We discuss related work in two areas. First, solving zero-sum games with exponentially large strategy sets. Efficient algorithms are known only for limited special cases. One approach is to represent the strategies in a lower dimensional space (the space of marginals). We elaborate more below since our algorithm uses this approach. For now, we just note that previous work (Ahmadinejad et al. 2016; Xu 2016; Chan et al. 2016) requires that the payoffs be linear.

We now introduce techniques our algorithm builds on.

Multilinear extension: We can view a set function \( f \) as being defined on the vertices of the hypercube \( \{0, 1\}^n \). Each vertex is the indicator vector of a set. A useful paradigm for submodular optimization is to extend \( f \) to a continuous function over \([0, 1]^n\) which agrees with \( f \) at the vertices. The multilinear extension \( F \) is defined as

\[
F(x) = \sum_{S \subseteq X} f(S) \prod_{j \in S} x_j \prod_{j \notin S} (1 - x_j).
\]

Equivalently, \( F(x) = \mathbb{E}_{S \sim p}[f(S)] \). That is, \( F(x) \) is the expected value of \( f \) on sets drawn from the independent distribution with marginals \( x \). \( F \) can be evaluated using random sampling (Calinescu et al. 2011) or in closed form for special cases (Iyer, Jegelka, and Bilmes 2014). Note that for any set \( S \) and its indicator vector \( 1_S \), \( F(1_S) = f(S) \). One crucial property of \( F \) is up-concavity (Calinescu et al. 2011). That is, \( F \) is concave along any direction \( u \geq 0 \) (where \( \geq \) denotes element-wise comparison). Formally, a function \( F \) is up-concave if for any \( x \) and any \( u \geq 0 \), \( F(x + \xi u) \) is concave as a function of \( \xi \).

Correlation gap: A useful property of submodular functions is that little is lost by optimizing only over independent distributions. Agrawal et al. (2010) introduced the concept of the correlation gap, which is the maximum ratio between the expectation of a function over an independent distribution and its expectation over a (potentially correlated) distribution with the same marginals. Let \( D(x) \) be the set of distributions with marginals \( x \). The correlation gap \( \kappa(f) \) of a function \( f \) is defined as

\[
\kappa(f) = \max_{x \in [0,1]^n} \max_{p \in D(x)} \frac{\mathbb{E}_{S \sim p}[f(S)]}{\mathbb{E}_{S \sim p}[f(S)]}.
\]

For any submodular function \( \kappa \leq \frac{e}{e-1} \). This says that, up to a loss of a factor \( 1 - 1/e \), we can restrict ourselves to independent distributions when solving Problem 1.

Swap rounding: Swap rounding is an algorithm developed by Chekuri et al. (2010) to round a fractional point in a matroid polytope to an integral point. We will use swap rounding to convert the fractional point obtained from the continuous optimization problem to a distribution over pure strategies. Swap rounding takes as input a representation of a point \( x \in \mathcal{P} \) as a convex combination of pure strategies. It then merges these sets together in a randomized fashion until only one remains. For any submodular function \( f \) and its multilinear extension \( F \), the random set \( R \) satisfies
The marginal space: A common meta-strategy for solving games with exponentially large strategy sets is to work in the lower-dimensional space of marginals. I.e., we keep track of only the marginal probability that each element in the ground set is chosen. To illustrate this, let \( p \) be a distribution over the pure strategies \( \mathcal{I} \), and \( x \in \mathcal{P} \) denote a vector giving the marginal probability of selecting each element of \( X \) in a set drawn according to \( p \). Note that \( x \) is \( n \)-dimensional while \( p \) could have dimension up to \( 2^n \). Previous work has used marginals for linear objectives. A linear function with weights \( w \) satisfies \( \mathbb{E}_{\mathcal{S}\sim p} \left[ \sum_{j \in S} w_j \right] = \sum_{j=1}^n w_j \mathbb{P}(j \in S) = \sum_{j=1}^n w_j x_j \), so keeping track of only the marginal probabilities \( x \) is sufficient for exact optimization. However, submodular functions do not in general satisfy this property: the utilities will depend on the full distribution \( p \), not just the marginals \( x \). We will treat a given marginal vector \( x \) as representing an independent distribution where each \( j \) is present with probability \( x_j \) (i.e., \( x \) compactly represents the full distribution \( p^x \)). The expected value of \( x \) under any sub modular function is exactly given by its multilinear extension, which is a continuous function.

Continuous extension: Let \( G = \min_i F_i \) be the pointwise minimum of the multilinear extensions of the functions in \( F \). Note that for any marginal \( x \), \( G(x) \) is exactly the objective value of \( p^x \) for Problem 1. Hence, optimizing \( G \) over all \( x \in \mathcal{P} \) is equivalent to solving Problem 1 restricted to independent distributions. Via the correlation gap, this restriction only loses a factor (1 – 1/e): if the optimal full distribution is \( p_{OPT} \), then the independent distribution with the same marginals as \( p_{OPT} \) has at least (1 – 1/e) of \( p_{OPT} \)‘s value under any submodular function. Previous algorithms (Calinescu et al. 2011; Bian et al. 2017) for optimizing up-concave functions like \( G \) do not apply because \( G \) is nonsmooth (see below). We introduce a novel Stochastic Frank-Wolfe algorithm which smooths the objective with random noise. Its runtime does not depend directly on \( |F| \) at all; it only uses BRI calls.

Rounding: Once we have solved the continuous problem, we need a way of mapping the resulting marginal vector \( x \) to a distribution over the pure strategies \( \mathcal{I} \). Notice that if we simply sample items independently according to \( x \), we might end up with an invalid set. For instance, in the uniform matroid which requires \(|S| \leq k \), an independent draw could result in more than \( k \) items even if \( \sum_i x_i \leq k \). Hence, we sample pure strategies by running the swap rounding algorithm on \( x \). In order to implement the maximizing player’s equilibrium strategy, it suffices to simply draw a sample whenever a decision is required. If a full description of the mixed strategy is desired, we show that it is sufficient to draw \( \Theta \left( \frac{1}{\epsilon} \log |F| + \log \frac{1}{\delta} \right) \) independent samples via swap rounding and return the uniform distribution over the sampled pure strategies.

Solving the continuous problem
The linchpin of our algorithmic strategy is solving the optimization problem \( \max_{x \in \mathcal{P}} G(x) \). In this section, we provide the ingredients to do so.

Properties of \( G \): We set the stage with four important properties of \( G \). First, while \( G \) is not in general concave, it is up-concave:

**Lemma 1.** If \( F_1 \ldots F_m \) are up-concave functions, then \( G = \min_i F_i \) is up-concave as well.

Up-concavity of \( G \) is the crucial property that enables efficient optimization.

Second, \( G \) is Lipschitz. Specifically, let \( M = \max_{i,j} f_i(j) \) be the maximum value of any single item. It can be shown that \( ||\nabla F_i||_\infty \leq M \forall i \) since (intuitively), the gradient of \( F_i \) is related to the marginal gain of items under \( f_i \). From this we derive

**Lemma 2.** \( G \) is \( M \)-Lipschitz in the \( \ell_1 \) norm.

Third, \( G \) is not smooth. For instance, it is not even differentiable at points where the minimizing function is not unique. This complicates the problem of optimizing \( G \) and renders earlier algorithms inapplicable.

Fourth, at any point \( x \) where the minimizing function \( F_i \) is unique, \( \nabla G(x) = \nabla F_i(x) \). Hence, we can compute \( \nabla G(x) \) by calling the BRI to find \( F_i \), and then computing \( \nabla F_i(x) \).

Algorithm 1: EQUATOR(\( BRI, FO, LO, u, c, K, r \))

1: \( x^0 \leftarrow u \)
2: //Stochastic Frank-Wolfe algorithm
3: for \( \ell = 1 \ldots K \) do
4: for \( t = 1 \ldots c \) do
5: \( z \sim \mu(u) \)
6: \( F_t \leftarrow BRI(x^{\ell-1} + z) \)
7: \( \nabla F_t \leftarrow FO(F_t, x^{\ell-1} + z) \)
8: end for
9: \( \nabla F^{\ell} \leftarrow \frac{1}{c} \sum_{t=1}^m \nabla F_t \)
10: \( v^{\ell} \leftarrow LO(\nabla F^{\ell}) \)
11: \( x^{\ell} \leftarrow x^{\ell-1} + \frac{1}{K} v^{\ell} \)
12: end for
13: \( x_{final} \leftarrow x^K - u \)
14: //Sample from equilibrium mixed strategy
15: Return \( r \) samples of SwapRound(\( x_{final} \))
In general, $\nabla F_\ell(x)$ can be computed by random sampling (Calinescu et al. 2011), and closed forms are known for particular cases (Iyer, Jegelka, and Bilmes 2014).

**Randomized smoothing:** We will solve the continuous problem $\max_{x \in P} G(x)$. Known strategies for optimizing up-concave functions (Bian et al. 2017) rely crucially on $G$ being smooth. Specifically, $\nabla G$ must be Lipschitz continuous. Unfortunately, $G$ is not even differentiable everywhere. Even between two points $x$ and $y$ where $G$ is differentiable, $\nabla G(x)$ and $\nabla G(y)$ can be arbitrarily far apart if $\arg \min_i F_i(x) \neq \arg \min_i F_i(y)$. No previous work addresses nonsmooth optimization of an up-concave function.

To resolve this issue, we use a carefully calibrated amount of random noise to smooth the objective. Let $\mu(u)$ be the uniform distribution over the $\ell_\infty$ ball of radius $u$. We define the smoothed objective $G_\mu(x) = \mathbb{E}_{z \sim \mu(u)} [G(x + z)]$ which averages over the region around $x$. This (and similar) techniques have been studied in the context of convex optimization (Duchi, Bartlett, and Wainwright 2012). We show that $G_\mu$ is a good smooth approximator of $G$.

**Lemma 3.** $G_\mu$ has the following properties:

- $G_\mu$ is up-concave.
- $|G_\mu(x) - G(x)| \leq \frac{Mn}{\mu}$, $\forall x$.
- $G_\mu$ is differentiable, with $\nabla G_\mu(x) = \mathbb{E}[\nabla G(x + z)]$.
- $\nabla G_\mu$ is $\frac{M}{\mu}$-Lipschitz continuous in the $\ell_1$ norm.

Hence, we can use $G_\mu$ as a better-behaved proxy for $G$ since it is both smooth and close to $G$ everywhere in the domain. The main challenge is that $G_\mu$ and its gradients are not available in closed form. Accordingly, we randomly sample values of the perturbation $z$ and average over the value of $G$ (or its gradient) at these sampled points.

**Stochastic Frank-Wolfe algorithm (SFW)**

We propose the SFW algorithm (Algorithm 1) to optimize $G_\mu$. SFW generates a series of feasible points $x^0, \ldots, x^K$, where $K$ is the number of iterations. Each point is generated from the last via two steps. First, SFW estimates the optimization oracle out these steps, SFW requires three oracles. First, a linear oracle is available in closed form. Accordingly, we randomly sample $w$ such that $E[w, v] = \mathbb{E}_{z \sim \mu(u)} [G(x + z)]$, which averages over the region around $x$. This (and similar) techniques have been studied in the context of convex optimization (Duchi, Bartlett, and Wainwright 2012). We show that $G_\mu$ is a good smooth approximator of $G$.

**Theoretical bounds**

Let $T_1$ be the runtime of the linear optimization oracle and $T_2$ be the runtime of the first-order oracle. We prove the following guarantee for SFW:

**Theorem 1.** For any $\epsilon, \delta > 0$, there are parameter settings such that SFW finds a solution $x^R$ satisfying $G(x^R) \geq (1 - \frac{1}{e}) \min_{u \in F} G(u) - \epsilon$ with probability at least $1 - \delta$. Its runtime is $\tilde{O}(T_1 M^2 K^2 n + T_2 k M^4 n^2 \log \frac{1}{\delta})$.

We remark that $T_1$ is small since linear optimization over $P$ can be carried out by a greedy algorithm. For instance, the runtime is $T_1 = \tilde{O}(n \log n)$ for the uniform matroid, which covers many applications. $T_2$ is typically dominated by the runtime of the BRI since it is known how to efficiently compute the gradient of a submodular function (Calinescu et al. 2011; Iyer, Jegelka, and Bilmes 2014).

Based on this result, we show the following guarantee on a single randomly sampled set that EQUATOR returns after applying swap rounding to the marginal vector $x_{final}$.

**Theorem 2.** With $r = 1$, EQUATOR outputs a set $S \in \mathcal{I}$ such that $\min_i \mathbb{E}[f_i(S)] \geq (1 - \frac{1}{e}) \min_{i \in F} \mathbb{E}[f_i(S)] - \epsilon$. Its time complexity is the same as SFW.

**Proof.** Suppose that $p_{OPT}$ is the distribution achieving the optimal value for Problem 1. Let $x^*$ be the optimizer for the problem $\max_{x \in P} G(x)$. That is, $x^*$ can be interpreted as the marginals of the independent distribution which maximizes $\min_i \mathbb{E}_{S \sim p^*_S}[f_i(S)]$. With slight abuse of notation, let $p_{OPT}$ be the independent distribution with the same marginals as $p_{OPT}$. By applying the correlation gap to each $f_i \in F$ and taking the min, we have

$$\min_{f_i \in F} \mathbb{E}_{S \sim p_{OPT}}[f_i(S)] \leq e - e - \min_{f_i \in F} \mathbb{E}_{S \sim p_{OPT}}[f_i(S)].$$

By definition of $x^*$, $G(x^*) \geq \min_{f_i \in F} \mathbb{E}_{S \sim p_{OPT}}[f_i(S)]$.

Hence, $G(x^*) \geq (1 - \frac{1}{e}) \min_{f_i \in F} \mathbb{E}_{S \sim p_{OPT}}[f_i(S)]$. Via Theorem 1, the marginal vector $x^*$ that our algorithm finds satisfies $G(x^*) \geq (1 - \frac{1}{e}) \min_{i \in F} \mathbb{E}[f_i(S)]$. Lastly, Chekuri et al. (2010) show that swap rounding outputs an independent set $S$ satisfying $\mathbb{E}[f_i(S)] \geq F_i(S)$ for any $f_i \in F$, which completes the proof.

\[\text{The } \tilde{O} \text{ notation hides logarithmic terms.}\]
This guarantee is sufficient if we just want to implement the maximizing player’s strategy by sampling an action. We also prove that if a full description of the maximizing player’s mixed strategy is desired, drawing a small number of independent samples via swap rounding suffices:

**Theorem 3.** Draw $r = \mathcal{O} \left( \frac{1}{\epsilon} \left( \log |F| + \log \frac{1}{\epsilon} \right) \right)$ samples using independent runs of randomized swap rounding. The uniform distribution on these samples is a $(1 - \frac{\epsilon}{2})^2 - \epsilon$ approximate equilibrium strategy for the maximizing player with probability at least $1 - \delta$. The runtime is $\mathcal{O} \left( \frac{rk^2 MF^2n}{\epsilon} \right)$.

**Applications**

We now give several examples of domains that our algorithm can be applied to. In each of these cases, we obtain the first guaranteed polynomial time constant-factor approximation algorithm for the problem. The key part of both applications is developing a BRI (the first order oracle is easily obtained in closed form via straightforward calculus).

**Network security games:** Earlier, we formulated network security games in the SBR framework. All we need to solve it using EQUATOR is a BRI oracle. The full attacker best response problem is known to be NP-hard (Jain et al. 2011). However, it turns out the best response to an independent distribution is easily computed. Index the set of paths and let $P_i$ be the $i$th path, ending at a target with value $\tau_i$. Let $P(t_j)$ be the set of all paths from the (super)source $s$ to $t_j$. Let $f_i$ be the corresponding submodular objective. Given a defender mixed strategy $x$, the attacker best response problem is to find $\min_i E_{S~\sim x}[f_i(S)]$. We can rewrite this as

$$\min_i E_{S~\sim x}[f_i(S)] = \min_i \mathbb{E} \left[ \tau_i \mathbf{1}[S \cap P_i \neq \emptyset] \right] = \min_{t_j \in T} \tau_j \min_{P \in P(t_j)} \mathbb{E} \left[ \mathbf{1}[S \cap P \neq \emptyset] \right] = \min_{t_j \in T} \tau_j \min_{P \in P(t_j)} 1 - \prod_{e \in P} (1 - x_e)$$

We can now solve a separate problem for each target $t_j$ and then take the one with lowest value. For each $t_j$, we solve a shortest path problem. We aim to find a $s - t_j$ path which maximizes the product of the weights of the path on each edge. Taking logarithms, this is equivalent to finding the path which minimizes $-\sum_{e \in P} \log(1 - x_e) = \sum_{e \in P} \log \frac{1}{1-x_e}$. This is a shortest path problem in which each edge has nonnegative weight $\log \frac{1}{1-x_e}$, and so can be solved via Dijkstra’s algorithm. With the attacker BRI in hand, applying EQUATOR yields the first subexponential-time algorithm for network security games.

**Robust coverage and budget allocation:** Many widespread applications of submodular functions concern coverage functions. A coverage function takes the following form. There a set of items $U$, and each $j \in U$ has a weight $w_j$. The algorithm can choose from a ground set $X = \{a_1, \ldots, a_n\}$ of actions. Each action $a_i$ covers a set $A_i \subseteq U$. The value of any set of actions is the total value of the items that those actions cover: $f(S) = \sum_{j \in \cup_{i \in S} A_i} w_j$. We can also consider probabilistic extensions where action $a_i$ covers each $j \in A_i$ independently with probability $p_{ij}$. This framework includes budget allocation, sensor placement, facility location, and many other common submodular optimization problems. Here we consider a robust coverage problem where the weights $w$ are unknown. For concreteness, we focus on the budget allocation problem, but all of our logic applies to general coverage functions.

Budget allocation models an advertiser’s choice of how to divide a finite budget $B$ between a set of advertising channels. Each channel is a vertex on the left hand side $L$ of a bipartite graph. The right hand $R$ consists of customers. Each customer $v \in R$ has a value $w_v$, which is the advertiser’s expected profit from reaching $v$. The advertiser allocates their budget in integer amounts among $L$. Let $y(s)$ denote the amount of budget allocated to channel $s \in L$. The advertiser solves the problem

$$\max_{y: \|y\|_1 \leq B} f_w(y) = \sum_{v \in R} w_v \left[ 1 - \prod_{s \in L} (1 - p_{sv})^{y(s)} \right]$$

where $p_{sv}$ is the probability that one unit of advertising on channel $s$ will reach customer $v$. This a probabilistic coverage problem where the action set $X$ contains $B$ copies of each $s \in L$ and the feasible decisions $I$ are all size $B$ subsets of $X$. Choosing $b$ copies of node $s$ corresponds to setting $y(s) = b$. Budget allocation has been the subject of a great deal of recent research (Alon, Gamzu, and Tennenholtz 2012; Soma et al. 2014; Miyachi et al. 2015).

In the robust optimization problem, the profits $w$ are not exactly known. Instead, they belong to a polyhedral uncertainty set $U$. This is very realistic: while an advertiser may be able to estimate the profit for each customer from past data, they are unlikely to know the true value for any particular campaign. We remark that Staib and Jegelka (2017) also considered a robust budget allocation problem, but their problem has uncertainty on the probabilities $p_{st}$, not the profits $w$. Further, they consider a continuous problem without the complication of rounding to discrete solutions.

As an example uncertainty set, consider the D-norm uncertain set, which is common in robust optimization (Bertsimas, Pachamanova, and Sim 2004; Staib and Jegelka 2017).

The uncertainty set is defined around a point estimate $\hat{\omega}$ as

$$U_{\hat{\omega}}^{D}\omega = \{\omega : \exists e \in [0, 1]|R|, w_i = (1 - c_i)\hat{\omega}_i, \|e\|_1 \leq \gamma\}$$

This can be thought of as allowing an adversary to scale down each entry of $\hat{\omega}$ with a total budget of $\gamma$. In our case, $\hat{\omega}$ is the advertiser’s best estimate from past data, and they would like to perform well for all scenarios within $U_{\hat{\omega}}^{D}\omega$. $\gamma$ defines the advertiser’s tolerance for risk. The problem we want to solve is $\max_{p \in \Delta(I)} \min_{\omega \in U_{\omega}} E_{\omega \sim p} f_{\omega}(y)$, which we recognize as an instance of Problem 1. For any fixed distribution $p$, we have by linearity of expectation

$$E_{\omega \sim p} f_{\omega}(y) = \sum_{w \in R} w_{\sim p} E_{\omega \sim p} \left[ 1 - \prod_{s \in L} (1 - p_{sv})^{y(s)} \right]$$

$^{2}$We use this formulation for simplicity, but it is possible to use only $\log B$ copies of each node (Ene and Nguyen 2016).
EQUATOR was run with $K = 100, c = 60, u = 0.1$.

Figure 1 shows the results. Figures 1(a) and 1(b) vary the network size $n$ with three randomly chosen source and target nodes. Figure 1(a) plots utility (i.e., how much loss is averted by the defender’s allocation) as a function of $n$. Error bars show one standard deviation. We see that EQUATOR obtains utility within 6% of SNARES, which computes a global optimum. Figure 1(b) shows runtime (on a logarithmic scale) as a function of $n$. SNARES was terminated after 10 hours for graphs with 250 nodes, while EQUATOR easily scales to 1000 nodes. Next, Figures 1(c) and 1(d) show results as the number of sources and targets grows. As expected, utility decreases with more sources/targets since the number of resources is constant and it becomes harder to defend the network. EQUATOR obtains utility within 4% of SNARES. However, SNARES was terminated after 10 hours for just 5 source/targets, while EQUATOR runs in under 25 seconds with 20 source/targets.

Robust budget allocation: We compare three algorithms for robust budget allocation. First, EQUATOR. Second, double oracle. We use the greedy algorithm for the defender’s best response (which is a $(1 − 1/e)$-approximation) since the exact best response is intractable. For the adversary’s best response, we use the linear program discussed in the section on robust coverage. Third, we compare to “greedy”, which greedily optimizes the advertiser’s return under the point estimate $w$. Greedy was implemented with lazy evaluation (Minoux 1978) which greatly improves its runtime at no cost to solution value. We generated random bipartite graphs with $|L| = |R| = n$ where each potential edge is present with probability 0.2 and for each edge $(u, v)$, $p_{u,v}$ is drawn uniformly in $[0, 0.2]$. $w$ was randomly generated with each coordinate uniform in $[0.5, 1.5]$. Our uncertainty set is the D-norm set around $w$ with $\gamma = \frac{1}{2}n$, representing a substantial degree of uncertainty. The budget was $B = 5 + 0.01 \cdot n$ since the problem is hardest when $B$ is small relative to $n$. EQUATOR was run with $K = 20, c = 10, u = 0.1$.

Figure 2 shows the results. Each point averages over 30 random problem instances (error bars would be hidden under the markers). Figure 2(a) plots the profit obtained by each algorithm when the true $w$ is chosen as the worst case...
in $\mathcal{U}^n$, with $n$ increasing on the $x$ axis. Figure 2(b) plots the average runtime for each $n$. We see that double oracle produces highly robust solutions. However, for even $n = 500$, its execution was halted after 10 hours. Greedy is highly scalable, but produces solutions that are approximately 40% less robust than double oracle. EQUATOR produces solution quality within 7% of double oracle and runs in less than 30 seconds with $n = 1000$.

Next, we show results on a real world dataset from Yahoo webscope (Yahoo 2007). The dataset logs bids placed by advertisers on a set of phrases. We create a budget allocation problem where the phrases are advertising channels and the accounts are targets; the resulting problem has $|L| = 1000$ and $|R| = 10,394$. Other parameters are the same as before. We obtain instances of varying size by randomly sampling a subset of $L$. Figures 2(c-d) show results (averaging over 30 random instances). In Figure 2(c), we see that both double oracle and EQUATOR find highly robust solutions, with EQUATOR’s solution value within 8% of that of double oracle. By contrast, greedy obtains no profit in the worst case for $|L| > 20$, validating the importance of robust solutions on real problems. In Figure 2(d), we observe that double oracle was terminated after 10 hours for $n = 500$ while EQUATOR scales to $n = 1000$ in under 40 seconds. We conclude that EQUATOR is empirically successful at finding highly robust solutions in an efficient manner, complementing its theoretical guarantees.

Acknowledgments: Wilder was supported by a NSF Graduate Fellowship and thanks Shaddin Dughmi for helpful conversations.

References

Agrawal, S.; Ding, Y.; Saberi, A.; and Ye, Y. 2010. Correlation robust stochastic optimization. In SODA.


Alon, N.; Gamzu, I.; and Tennenholtz, M. 2012. Optimizing budget allocation among channels and influencers. In EC.

Agrawal, S.; Ding, Y.; Saberi, A.; and Ye, Y. 2010. Correlation robust stochastic optimization. In SODA.


Alon, N.; Gamzu, I.; and Tennenholtz, M. 2012. Optimizing budget allocation among channels and influencers. In EC.

Agrawal, S.; Ding, Y.; Saberi, A.; and Ye, Y. 2010. Correlation robust stochastic optimization. In SODA.


Alon, N.; Gamzu, I.; and Tennenholtz, M. 2012. Optimizing budget allocation among channels and influencers. In EC.

Agrawal, S.; Ding, Y.; Saberi, A.; and Ye, Y. 2010. Correlation robust stochastic optimization. In SODA.


Alon, N.; Gamzu, I.; and Tennenholtz, M. 2012. Optimizing budget allocation among channels and influencers. In EC.

Agrawal, S.; Ding, Y.; Saberi, A.; and Ye, Y. 2010. Correlation robust stochastic optimization. In SODA.


Alon, N.; Gamzu, I.; and Tennenholtz, M. 2012. Optimizing budget allocation among channels and influencers. In EC.

Agrawal, S.; Ding, Y.; Saberi, A.; and Ye, Y. 2010. Correlation robust stochastic optimization. In SODA.


Alon, N.; Gamzu, I.; and Tennenholtz, M. 2012. Optimizing budget allocation among channels and influencers. In EC.

Agrawal, S.; Ding, Y.; Saberi, A.; and Ye, Y. 2010. Correlation robust stochastic optimization. In SODA.


Alon, N.; Gamzu, I.; and Tennenholtz, M. 2012. Optimizing budget allocation among channels and influencers. In EC.

Agrawal, S.; Ding, Y.; Saberi, A.; and Ye, Y. 2010. Correlation robust stochastic optimization. In SODA.


Alon, N.; Gamzu, I.; and Tennenholtz, M. 2012. Optimizing budget allocation among channels and influencers. In EC.

Agrawal, S.; Ding, Y.; Saberi, A.; and Ye, Y. 2010. Correlation robust stochastic optimization. In SODA.


Alon, N.; Gamzu, I.; and Tennenholtz, M. 2012. Optimizing budget allocation among channels and influencers. In EC.

Agrawal, S.; Ding, Y.; Saberi, A.; and Ye, Y. 2010. Correlation robust stochastic optimization. In SODA.