The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

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 either heuristics 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 ϵ) 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, nonsmooth 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 $f_1 ... f_m$. 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 algorithm 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 and easily scales to instances that are too large for previous algorithms to handle.

All proofs can be found in the full version of the paper (Wilder 2017), which also makes two further contributions. First, we extend our techniques to obtain bicriteria guarantees for deterministic robust submodular optimization. Second, we show that when a stronger form of best response oracle is available for the adversary, EQUATOR's approximation guarantee can be improved to $(1 - 1/e)$.

Problem description

Formulation: Let X be a set of items with $|X| = n$. A function $f : 2^X \to R$ is submodular if for any $A \subseteq B$ and $i \in X \setminus B$, $f(A \cup \{i\}) - f(A) \geq f(B \cup \{i\}) - f(B)$. We restrict our attention to functions that are *monotone*, i.e., $f(A \cup \{i\}) - f(A) \geq 0$ for all $i \in X, A \subset X$. Without loss of generality, we assume that $f(\emptyset) = 0$ and hence $f(S) \geq 0\, \forall S$. Let $\mathcal{F} = \{f_1 ... f_m\}$ be a finite set of submodular functions on the ground set X. m may be exponentially large. Let $\Delta(S)$ denote the set of probability distributions over the elements of any set S. Oftentimes, we will work with independent distributions over X, which can be fully specified by a vector $\boldsymbol{x} \in R^n_+$. x_i gives the marginal probability that item i is chosen. Denote by $p^I_{\boldsymbol{x}}$ the independent distribution with marginals $\boldsymbol{x}$. Let $\mathcal{I}$ be a collection of subsets of X. For instance, we could have $\mathcal{I} = \{S \subseteq X : |S| \leq k\}$. We would like to find a minimax equilibrium of the game where the maximizing player's pure strategies are the subsets in $\mathcal{I}$, and the minimizing player's pure strategies are the functions in $\mathcal{F}$. The payoff to the strategies $S \in \mathcal{I}$ and $f_i \in \mathcal{F}$ is $f_i(S)$. We call a game in this form a *submodular best response* (SBR) game. For the maximizing player, computing the minimax equilibrium is equivalent to solving

$$\max_{p \in \Delta(\mathcal{I})} \min_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{S \sim p} [f(S)] \tag{1}$$

where $S \sim p$ denotes that S is distributed according to p.

Example: network security games. To make the setting more concrete, we now introduce one of our example domains, the network security game of Tsai et al. (2010). There is a graph $G = (V, E)$. There is a source vertex s (which may be a supersource connected to multiple real sources) and a set of targets T. An attacker wishes to traverse the network starting from the source and attack a target. Each target t_j has a value τ_j. The attacker picks a $s - t_j$ path for some $t_j \in T$. The defender attempts to catch the attacker by protecting edges of the network. The defender may select any k edges, and the attacker is caught if any of these edges lies on the chosen path. We use the normalized utilities defined by Jain et al. (2013), which give the defender utility $\tau_j > 0$ if an attack on t_j is intercepted and 0 if the attack succeeds. Thus, each path P from s to t_j for the attacker induces an objective function f_P for the defender: for any set of edges S, $f_P(S) = \tau_j$ if $S \cap P \neq \emptyset$, otherwise $f_P(S) = 0$. f_P is easily seen to be submodular (Jain, Conitzer, and Tambe 2013). Hence, we have a SBR game with $\mathcal{I} = \{S \subseteq E : |S| \leq k\}$ and $\mathcal{F} = \{f_P : P \text{ is a path from } s \text{ to } T\}$.

Allowable pure strategy sets: Our running example is when the pure strategies $\mathcal{I}$ of the maximizing player are all size k subsets: $\mathcal{I} = \{S \subseteq X : |S| \leq k\}$. In general, our algorithm works when $\mathcal{I}$ is any matroid; this example is called the *uniform* matroid. We refer to Korte and Vygen (2012) for more details on matroids. Here, we just note that matroids are a class of well-behaved constraint structures which are of great interest in combinatorial optimization. A useful fact is that any linear objective can be exactly optimized over a matroid by the greedy algorithm. For instance, consider the above uniform matroid. If each element j has a weight w_j, the highest weighted set of size k is obtained simply by taking the k items with highest individual weights. Let $k = \max_{S \in \mathcal{I}} |S|$ be the size of the largest pure strategy. E.g., in network security games k is the number of defender resources. In general, k is the rank of the matroid.

We now introduce some notation for the continuous extension of the problem. Let $\mathbf{1}_S$ be the indicator vector of the set S (i.e., an n-dimensional vector with 1 in the entries of elements that are in S and 0 elsewhere). Let $\mathcal{P}$ be the convex hull of $\{\mathbf{1}_S : S \in \mathcal{I}\}$. Note that $\mathcal{P}$ is a polytope.

Best response oracles: A best response oracle for one player is a subroutine which computes the pure strategy with highest expected utility against a mixed strategy for the other player. We assume that an oracle is available for the minimzing player. However, we require only a weaker oracle, which we call an *best response to independent distributions oracle* (BRI). A BRI oracle is only required to compute a best response to mixed strategies which are independent distributions, represented as the marginal probability that each item in X appears. Given a vector $\boldsymbol{x} \in R^n_+$, where x_i is the probability that element $i \in X$ is chosen, a BRI oracle computes $\arg\min_{f_i \in \mathcal{F}} \mathbb{E}_{S \sim p^I_{\boldsymbol{x}}} [f_i(S)]$. We use $S \sim \boldsymbol{x}$ to denote that S is drawn from the independent distribution with marginals $\boldsymbol{x}$. As we will see later, sometimes a BRI oracle is readily available even when the full best response is NP-hard.

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 Udwani 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* in the lower dimensional space. Linearity is a very restrictive assumption; ours is the first algorithm which extends the marginal-based approach to general submodular functions. This requires entirely different techniques.

In practice, large zero sum games are often solved via the *double oracle* algorithm (McMahan, Gordon, and Blum 2003; Bosansky et al. 2014; Halvorson, Conitzer, and Parr 2009). Double oracle starts with each player restricted to only a small number of pure strategies and repeatedly adds a new strategy for each player until an equilibrium is reached. The new strategies are chosen to be each player's best response to the other's current mixed strategy. This technique is appealing when equilibria have sparse support, and so only a few iterations are needed. However, it is easy to give examples where *every* pure strategy lies in the support of the equilibrium, so double oracle will require exponential runtime. Our algorithm runs in guaranteed polynomial time.

Second, we give more background on robust submodular optimization. Krause et al. (2008) introduced the problem of maximizing the minimum of submodular functions, which corresponds to Problem 1 with the maximizing player restricted to pure strategies. They show that the problem is inapproximable unless P = NP. They then relax the problem by allowing the algorithm to exceed the budget constraint (a bicriteria guarantee). Our primary focus is on the *randomized* setting, where the algorithm respects the budget constraint but chooses a distribution over actions instead of a pure strategy. This randomized variant was studied by Wilder et al. (2017) for the special case of influence maximization. Krause et al. (2011) and Chen et al. (2017) studied general submodular functions using very similar techniques: both iterate dynamics where the adversary plays a no-regret learning algorithm and the decision maker plays a greedy best response. This algorithm maintains a variable for every function in $\mathcal{F}$ and so is only computationally tractable when $\mathcal{F}$ is small. By contrast, we deal with the setting where $\mathcal{F}$ is exponentially large. However, we lose an extra factor of $(1 - 1/e)$ in the approximation ratio.

In the full version of the paper, we extend our algorithm to obtain bicriteria guarantees for the deterministic problem (where we select a single feasible set). Our guarantees apply under significantly more general conditions than those of Krause et al. (2008): our algorithm generalizes from cardinality constraints to arbitrary matroid constraints and only needs BRI oracle access to $\mathcal{F}$ instead of explicitly enumerating the objectives. The size of the set returned scales as $\log |\mathcal{F}|$, which is the optimal dependence on $|\mathcal{F}|$ under complexity-theoretic assumptions (Krause et al. 2008). However, the size also depends on a precision parameter ϵ.

Preliminaries

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

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

Equivalently, $F(\boldsymbol{x}) = \mathbb{E}_{S \sim \boldsymbol{x}}[f(S)]$. That is, $F(\boldsymbol{x})$ is the expected value of f on sets drawn from the independent distribution with marginals $\boldsymbol{x}$. 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 $\mathbf{1}_S$, $F(\mathbf{1}_S) = f(S)$. One crucial property of F is *up-concavity* (Calinescu et al. 2011). That is, F is concave along any direction $\boldsymbol{u} \succeq \mathbf{0}$ (where $\succeq$ denotes element-wise comparison). Formally, a function F is *up-concave* if for any $\boldsymbol{x}$ and any $\boldsymbol{u} \succeq 0$, $F(\boldsymbol{x} + \xi \boldsymbol{u})$ is concave as a function of ξ.

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 $D(\boldsymbol{x})$ be the set of distributions with marginals $\boldsymbol{x}$. The correlation gap $\kappa(f)$ of a function f is defined as

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

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 $\boldsymbol{x} \in \mathcal{P}$ 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

$\mathbb{E}[f(R)] \geq F(\boldsymbol{x})$. I.e., swap rounding only increases the value of any submodular function in expectation.

Algorithm for SBR games

In this section, we introduce the EQUATOR (*EQUilibrium via stochAsTic frank-wOlfe and Rounding*) algorithm for computing approximate equilibrium strategies for the maximizing player in SBR games. Since the pure strategy sets can be exponentially large, it is unclear what it even means to compute an equilibrium: representing a mixed strategy may require exponential space. Our solution to this dilemma is to show how to efficiently *sample* pure strategies from an approximate equilibrium mixed strategy. This suffices for the maximizing player to implement their strategy. Alternatively, we can build an approximate mixed strategy with sparse support by drawing a polynomial number of samples and outputing the uniform distribution over the samples. In order to generate these samples, EQUATOR first solves a continuous optimization problem, which we now describe.

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 $\boldsymbol{x} \in \mathcal{P}$ denote a vector giving the marginal probability of selecting each element of X in a set drawn according to p. Note that $\boldsymbol{x}$ 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 $\mathbb{E}_{S\sim p}\left[\sum_{j\in S} w_j\right] = \sum_{j=1}^{n} w_j \text{Pr}[j \in S] = \sum_{j=1}^{n} w_j x_j$, so keeping track of only the marginal probabilities $\boldsymbol{x}$ 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 $\boldsymbol{x}$. We will treat a given marginal vector $\boldsymbol{x}$ as representing an independent distribution where each j is present with probability x_j (i.e., $\boldsymbol{x}$ compactly represents the full distribution $p^I_{\boldsymbol{x}}$). The expected value of $\boldsymbol{x}$ under any submodular 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 $\mathcal{F}$. Note that for any marginal $\boldsymbol{x}$, $G(\boldsymbol{x})$ is exactly the objective value of $p^I_{\boldsymbol{x}}$ for Problem 1. Hence, optimizing G over all $\boldsymbol{x} \in \mathcal{P}$ 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 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 $|\mathcal{F}|$ 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 $\boldsymbol{x}$ to a distribution over the pure strategies $\mathcal{I}$. Notice that if we simply sample items independently according to $\boldsymbol{x}$, 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 $\boldsymbol{x}$. 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 $\Theta\left(\frac{1}{\epsilon^3}(\log|\mathcal{F}| + \log\frac{1}{\delta})\right)$ independent samples via swap rounding and return the uniform distribution over the sampled pure strategies.

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

```
1:  $\boldsymbol{x}^0 \leftarrow u\mathbf{1}$
2:  //Stochastic Frank-Wolfe algorithm
3:  for $\ell = 1...K$ do
4:      for $t = 1...c$ do
5:          Draw $\boldsymbol{z} \sim \mu(u)$
6:          $F_t \leftarrow$ BRI$(\boldsymbol{x}^{\ell-1} + \boldsymbol{z})$
7:          $\tilde{\nabla}^\ell_t \leftarrow FO(F_t, \boldsymbol{x}^{\ell-1} + \boldsymbol{z})$
8:      end for
9:      $\tilde{\nabla}^\ell \leftarrow \frac{1}{c}\sum_{t=1}^{m} \tilde{\nabla}^\ell_t$
10:     $\boldsymbol{v}^\ell \leftarrow LO(\tilde{\nabla}^\ell)$
11:     $\boldsymbol{x}^\ell \leftarrow \boldsymbol{x}^{\ell-1} + \frac{1}{K}\boldsymbol{v}^\ell$
12: end for
13: $\boldsymbol{x}_{final} \leftarrow \boldsymbol{x}^K - u\mathbf{1}$
14: //Sample from equilibrium mixed strategy
15: Return $r$ samples of SwapRound$(\boldsymbol{x}_{final})$
```

Solving the continuous problem

The linchpin of our algorithmic strategy is solving the optimization problem $\max_{\boldsymbol{x}\in\mathcal{P}} G(\boldsymbol{x})$. 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 $F_1...F_m$ 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 ℓ_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 $\boldsymbol{x}$ where the minimizing function F_i is unique, $\nabla G(\boldsymbol{x}) = \nabla F_i(\boldsymbol{x})$. Hence, we can compute $\nabla G(\boldsymbol{x})$ by calling the BRI to find F_i, and then computing $\nabla F_i(\boldsymbol{x})$.

In general, $\nabla F_i(\boldsymbol{x})$ 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 $\max_{\boldsymbol{x}\in\mathcal{P}} G(\boldsymbol{x})$. Known strategies for optimizing up-concave functions (Bian et al. 2017) rely crucially on G being smooth. Specifically, ∇G must be Lipschitz continuous. Unfortunately, G is not even differentiable everywhere. Even between two points $\boldsymbol{x}$ and $\boldsymbol{y}$ where G is differentiable, $\nabla G(\boldsymbol{x})$ and $\nabla G(\boldsymbol{y})$ can be arbitrarily far apart if $\arg\min_i F_i(\boldsymbol{x}) \neq \arg\min_i F_i(\boldsymbol{y})$. 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 ℓ_∞ ball of radius u. We define the smoothed objective $G_\mu(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}\sim\mu(u)}\left[G(\boldsymbol{x}+\boldsymbol{z})\right]$ which averages over the region around $\boldsymbol{x}$. This (and similar) techniques have been studied in the context of convex optimization (Duchi, Bartlett, and Wainwright 2012). We show that G_μ is a good smooth approximator of G.

Lemma 3. *G_μ has the following properties:*

- *G_μ is up-concave.*
- $|G_\mu(\boldsymbol{x}) - G(\boldsymbol{x})| \leq \frac{Mnu}{2} \quad \forall \boldsymbol{x}$.
- *G_μ is differentiable, with $\nabla G_\mu(x) = \mathbb{E}[\nabla G(\boldsymbol{x}+\boldsymbol{z})]$.*
- *∇G_μ is $\frac{M}{\mu}$−Lipschitz continuous in the ℓ_1 norm.*

Hence, we can use G_μ 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_μ and its gradients are not available in closed form. Accordingly, we randomly sample values of the perturbation $\boldsymbol{z}$ 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_μ. SFW generates a series of feasible points $\boldsymbol{x}^0 ... \boldsymbol{x}^K$, where K is the number of iterations. Each point is generated from the last via two steps. First, SFW estimates the gradient of G_μ. Second, it takes a step towards the point in $\mathcal{P}$ which is furthest in the direction of the gradient. To carry out these steps, SFW requires three oracles. First, a linear optimization oracle LO which, given an objective $\boldsymbol{w}$, returns $\arg\max_{\boldsymbol{v}\in\mathcal{P}} \boldsymbol{w}^\top \boldsymbol{v}$. In the context of our problem, LO outputs the indicator vector of the set $S \in \mathcal{I}$ which maximizes the linear objective $\boldsymbol{w}$. S can be efficiently found via the greedy algorithm. The other two oracles concern gradient evaluation. One is the BRI oracle discussed earlier. The other is a stochastic first-order oracle FO which, for any function F_i and point $\boldsymbol{x}$, returns an unbiased estimate of $\nabla F_i(\boldsymbol{x})$.

The algorithm starts at $\boldsymbol{x}^0 = \boldsymbol{0}$. At each iteration ℓ, it averages over c calls to FO to compute a stochastic approximation $\tilde{\nabla}^\ell$ to $\nabla G_\mu(\boldsymbol{x}^{\ell-1})$ (Lines 4-9). For each call, it draws a random perturbation $\boldsymbol{z} \sim \mu(u)$ and uses the BRI to find the minimizing F at $\boldsymbol{x}^{\ell-1} + \boldsymbol{z}$. It then queries FO for an estimate of $\nabla F(\boldsymbol{x}^{\ell-1} + \boldsymbol{z})$. Lastly, it takes a step in the direction of $\boldsymbol{v}^\ell = LO(\tilde{\nabla}^\ell)$ by setting $\boldsymbol{x}^\ell = \boldsymbol{x}^{\ell-1} + \frac{1}{K}\boldsymbol{v}^\ell$ (Lines 10-11). Since $\boldsymbol{x}^\ell$ at each iteration is a combination of vertices of $\mathcal{P}$, the output is guaranteed to be feasible. The intuition for why the algorithm succeeds is that it only moves along nonnegative directions (since $\boldsymbol{v}^\ell$ is always nonnegative). This is in contrast to gradient-based algorithms for *concave* optimization, which move in the (possibly negative) direction $\boldsymbol{v}^\ell - \boldsymbol{x}^\ell$. As an up-concave function, G_μ is concave along all nonnegative directions. By moving only in such directions we inherit enough of the nice properties of concave optimization to obtain a $(1 - 1/e)-$ approximation.

A small technical detail is that adding random noise $\boldsymbol{z}$ could result in negative values, for which the multilinear extension is not defined. To circumvent this, we start the algorithm at $\boldsymbol{x}^0 = u\boldsymbol{1}$ (i.e., with small positive values in every entry) and then return $\boldsymbol{x}_{final} = \boldsymbol{x}^K - u\boldsymbol{1}$ (Line 13).

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 $\boldsymbol{x}^K$ satisfying $G(\boldsymbol{x}^K) \geq (1 - \frac{1}{e})OPT - \epsilon$ with probability at least $1-\delta$. Its runtime is $\tilde{\mathcal{O}}\left(T_1 \frac{M^2 k^2 n}{\epsilon^2} + T_2 \frac{k^4 M^4 n}{\epsilon^4} \log \frac{1}{\delta}\right)$*[1].

We remark that T_1 is small since linear optimization over $\mathcal{P}$ can be carried out by a greedy algorithm. For instance, the runtime is $T_1 = \mathcal{O}(n \log n)$ 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 $\boldsymbol{x}_{final}$.

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

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

$$\min_{f_i\in\mathcal{F}} \mathop{\mathbb{E}}_{S\sim p_{OPT}}[f_i(S)] \leq \frac{e}{e-1} \min_{f_i\in\mathcal{F}} \mathop{\mathbb{E}}_{S\sim p^I_{OPT}}[f_i(S)].$$

By definition of $\boldsymbol{x}^*$, $G(\boldsymbol{x}^*) \geq \min_{f_i\in\mathcal{F}} \mathbb{E}_{S\sim p^I_{OPT}}[f_i(S)]$. Hence, $G(\boldsymbol{x}^*) \geq (1 - 1/e)\min_i \mathbb{E}_{S\sim p^I_{\boldsymbol{x}^*}}[f_i(S)] = (1 - 1/e)OPT$. Via Theorem 1, the marginal vector $\boldsymbol{x}$ that our algorithm finds satisfies $G(\boldsymbol{x}) \geq (1 - \frac{1}{e})G(\boldsymbol{x}^*) - \epsilon \geq (1 - \frac{1}{e})^2 OPT - \epsilon$. 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 $f_i \in \mathcal{F}$, which completes the proof. $\square$

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

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 ith path, ending at a target with value τ_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 $\boldsymbol{x}$, the attacker best response problem is to find $\min_i \mathbb{E}_{S\sim\boldsymbol{x}}[f_i(S)]$. We can rewrite this as

$$\begin{aligned}\min_i \mathop{\mathbb{E}}_{S\sim\boldsymbol{x}}[f_i(S)] &= \min_i \mathop{\mathbb{E}}_{S\sim\boldsymbol{x}}[\tau_i \mathbf{1}[S\cap P_i \neq \emptyset]]\\ &= \min_{t_j\in T}\tau_j \min_{P\in P(t_j)} \mathop{\mathbb{E}}_{S\sim\boldsymbol{x}}[\mathbf{1}[S\cap P\neq\emptyset]]\\ &= \min_{t_j\in T}\tau_j \min_{P\in P(t_j)} 1 - \prod_{e\in P}[1-x_e]\end{aligned}$$

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 the weights $1 - x_e$ 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 $X = \{a_1...a_n\}$ 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: $f(S) = \sum_{j\in\bigcup_{i\in S}A_i} w_j$. 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 $\boldsymbol{w}$ 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

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

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[2] of each $s \in L$ and the feasible decisions $\mathcal{I}$ 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; Miyauchi et al. 2015).

In the robust optimization problem, the profits $\boldsymbol{w}$ are not exactly known. Instead, they belong to a polyhedral uncertainty set $\mathcal{U}$. 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 $\boldsymbol{w}$. 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 $\hat{\boldsymbol{w}}$ as

$$\mathcal{U}_\gamma^{\hat{\boldsymbol{w}}} = \{\boldsymbol{w} : \exists \boldsymbol{c}\in[0,1]^{|R|}, w_i = (1-c_i)\hat{w}_i,\ ||\boldsymbol{c}||_1 \le \gamma\}.$$

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

$$\mathop{\mathbb{E}}_{\boldsymbol{y}\sim p}[f_{\boldsymbol{w}}(\boldsymbol{y})] = \sum_{v\in R} w_v \mathop{\mathbb{E}}_{\boldsymbol{y}\sim p}\left[1 - \prod_{s\in L}(1-p_{sv})^{y(s)}\right].$$

[2]We use this formulation for simplicity, but it is possible to use only $\log B$ copies of each node (Ene and Nguyen 2016).

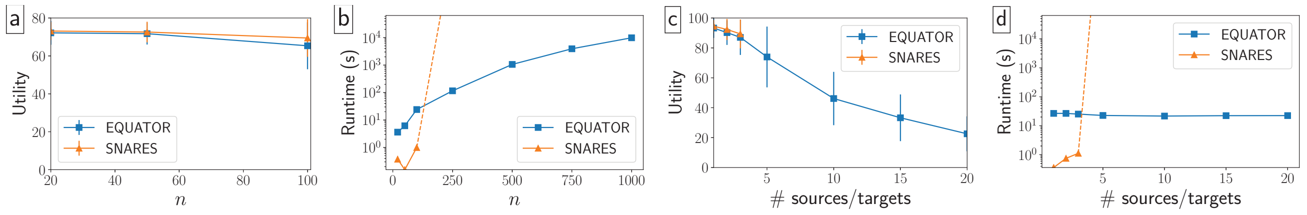

Figure 1: Experimental results for network security games.

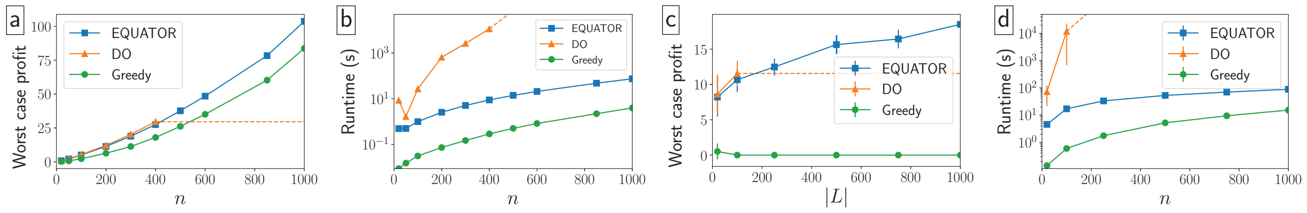

Figure 2: Experimental results for budget allocation.

Note that the inner expectation (which is the total probability that each $v \in R$ is reached) is constant with respect to $\boldsymbol{w}$. Hence, the adversary's best response problem of computing $\min_{\boldsymbol{w} \in \mathcal{U}} \mathbb{E}_{\boldsymbol{y} \sim p}[f_{\boldsymbol{w}}(\boldsymbol{y})]$ is a linear program and can be easily solved. We remark that the coefficients of this LP (the inner expectation in the above sum) can easily be computed exactly for any independent distribution. Further, since any LP has an optimal solution among the vertices of $\mathcal{U}_\gamma^{\hat{\boldsymbol{w}}}$, we can without loss of generality restrict the adversary's pure strategies to a finite (though exponentially large) number.

Experiments

We now show experimental results from applying EQUATOR to two different domains.

Network security games: We first study the network security game defined above. We compare EQUATOR to the SNARES algorithm (Jain, Conitzer, and Tambe 2013) which is the current state of the art algorithm with guaranteed solution quality. SNARES uses a double oracle approach to find a *globally* optimal solution. However, it incorporates several domain-specific heuristics which substantially improve its runtime over a standard implementation of double oracle. We note that Iwashita et al. (2016) proposed a newer double-oracle style algorithm which first preprocesses the graph to remove unnecessary edges. We do not compare to this approach because the preprocessing step can be applied equally well to either EQUATOR or double oracle. We use random geometric graphs, which are commonly used to assess algorithms for this domain due to their similarity to real world road networks (Jain, Conitzer, and Tambe 2013; Iwashita et al. 2016). As in Jain et al. (2013), we use density $d = 0.1$ with the value of each target drawn uniformly at random in $[0, 100]$. We set k to be one percent of the number of edges. Each data point averages over 30 random instances. 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 $\hat{\boldsymbol{w}}$. 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 draw uniformly in $[0, 0.2]$. $\hat{\boldsymbol{w}}$ was randomly generated with each coordinate uniform in $[0.5, 1.5]$. Our uncertainty set is the D-norm set around $\hat{\boldsymbol{w}}$ 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 $\boldsymbol{w}$ is chosen as the worst case in $\mathcal{U}_\gamma^{\hat{\mathbf{w}}}$, 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*.

Ahmadinejad, A.; Dehghani, S.; Hajiaghayi, M.; Lucier, B.; Mahini, H.; and Seddighin, S. 2016. From duels to battlefields: Computing equilibria of Blotto and other games. In *AAAI*.

Alon, N.; Gamzu, I.; and Tennenholtz, M. 2012. Optimizing budget allocation among channels and influencers. In *WWW*, 381–388.

Bertsimas, D.; Pachamanova, D.; and Sim, M. 2004. Robust linear optimization under general norms. *Operations Research Letters* 32(6):510–516.

Bian, A. A.; Mirzasoleiman, B.; Buhmann, J. M.; and Krause, A. 2017. Guaranteed non-convex optimization: Submodular maximization over continuous domains. In *AISTATS*.

Bosansky, B.; Kiekintveld, C.; Lisy, V.; and Pechoucek, M. 2014. An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. *Journal of Artificial Intelligence Research* 51:829–866.

Calinescu, G.; Chekuri, C.; Pál, M.; and Vondrák, J. 2011. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing* 40(6):1740–1766.

Chan, H.; Jiang, A. X.; Leyton-Brown, K.; and Mehta, R. 2016. Multilinear games. In *WINE*.

Chekuri, C.; Vondrak, J.; and Zenklusen, R. 2010. Dependent randomized rounding via exchange properties of combinatorial structures. In *FOCS*.

Chen, R.; Lucier, B.; Singer, Y.; and Syrgkanis, V. 2017. Robust optimization for non-convex objectives. In *NIPS*.

Delage, E.; Kuhn, D.; and Wiesemann, W. 2016. dice-sion making under uncertainty: When can a random decision reduce risk? Technical Report EPFL-ARTICLE-220662.

Duchi, J. C.; Bartlett, P. L.; and Wainwright, M. J. 2012. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization* 22(2):674–701.

Ene, A., and Nguyen, H. L. 2016. A reduction for optimizing lattice submodular functions with diminishing returns. *arXiv preprint arXiv:1606.08362*.

Halvorson, E.; Conitzer, V.; and Parr, R. 2009. Multi-step multi-sensor hider-seeker games. In *IJCAI*.

Iwashita, H.; Ohori, K.; Anai, H.; and Iwasaki, A. 2016. Simplifying urban network security games with cut-based graph contraction. In *AAMAS*.

Iyer, R. K.; Jegelka, S.; and Bilmes, J. A. 2014. Monotone closure of relaxed constraints in submodular optimization: Connections between minimization and maximization. In *UAI*.

Jain, M.; Korzhyk, D.; Vaněk, O.; Conitzer, V.; Pěchouček, M.; and Tambe, M. 2011. A double oracle algorithm for zero-sum security games on graphs. In *AAMAS*.

Jain, M.; Conitzer, V.; and Tambe, M. 2013. Security scheduling for real-world networks. In *AAMAS*.

Korte, B., and Vygen, J. 2012. *Combinatorial optimization*. Springer.

Krause, A.; McMahan, H. B.; Guestrin, C.; and Gupta, A. 2008. Robust submodular observation selection. *Journal of Machine Learning Research* 9(Dec):2761–2801.

Krause, A.; Roper, A.; and Golovin, D. 2011. Randomized sensing in adversarial environments. In *IJCAI*.

McMahan, H. B.; Gordon, G. J.; and Blum, A. 2003. Planning in the presence of cost functions controlled by an adversary. In *ICML*.

Minoux, M. 1978. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques* 234–243.

Miyauchi, A.; Iwamasa, Y.; Fukunaga, T.; and Kakimura, N. 2015. Threshold influence model for allocating advertising budgets. In *ICML*, 1395–1404.

Orlin, J. B.; Schulz, A. S.; and Udwani, R. 2016. Robust monotone submodular function maximization. In *IPCO*.

Soma, T.; Kakimura, N.; Inaba, K.; and Kawarabayashi, K.-i. 2014. Optimal budget allocation: Theoretical guarantee and efficient algorithm. In *ICML*, 351–359.

Staib, M., and Jegelka, S. 2017. Robust budget allocation via continuous submodular functions. In *ICML*.

Tsai, J.; Yin, Z.; Kwak, J.-y.; Kempe, D.; Kiekintveld, C.; and Tambe, M. 2010. Urban security: Game-theoretic resource allocation in networked physical domains. In *National Conference on Artificial Intelligence (AAAI)*.

Wilder, B.; Yadav, A.; Immorlica, N.; Rice, E.; and Tambe, M. 2017. Uncharted but not uninfluenced: Influence maximization with an uncertain network. In *AAMAS*.

Wilder, B. 2017. Equilibrium computation and robust optimization in zero sum games with submodular structure. *arXiv preprint arXiv:1710.00996*.

Xu, H. 2016. The mysteries of security games: Equilibrium computation becomes combinatorial algorithm design. In *EC*.

Yahoo. 2007. Yahoo! webscope dataset ydata-ysm-advertiser-bids-v1 0. http://research.yahoo.com/Academic_Relations.