Equilibrium computation for 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 \((1 - \frac{1}{e})^2\)-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 Jain et al. (2011). 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 \(\epsilon\)) 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 extension of the problem and then rounds the solution back 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 (see Krause and Golovin (2014) for a survey).
One prominent class of applications 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 \(\min S f_i\). Robust submodular optimization has many applications because uncertainty is so often present in decision-making. We study 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 \cite{KrauseRoperGolovin2011, WiesemannKuhnRustem2013, Wilder2017}. This is equivalent to computing the minimax equilibrium for a game where one player has a submodular best response.

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 \cite{Jain2011} 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.

**Problem description**

**Formulation:** Let \(X\) be a set of items with \(|X| = n\). A function \(f : 2^X \to \mathbb{R}\) 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 \(i \in X, A \subseteq X\). Without loss of generality, we assume that \(f(\emptyset) = 0\) and hence \(f(S) \geq 0 \forall S\). Let \(\mathcal{F} = \{f_1, \ldots, f_m\}\) 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 \(x \in \mathbb{R}^n_+\). \(x_i\) gives the marginal probability that item \(i\) is chosen. Denote by \(p_x\) the independent distribution with marginals \(x\). 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 \(f_j \in \mathcal{F}\) is \(f_j(S)\). 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

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

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 Jain et al. \cite{Jain2011}. 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 \(\tau_j\). The attacker picks a path from \(s\) to \(t_j\) 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. \cite{Jain2013}, 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 \cite{Jain2013}. 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: \(\mathcal{I} = \{X : |X| \leq k\}\). In general, our algorithm works when \(\mathcal{I}\) is any matroid; this example is called the uniform matroid. We refer to \cite{Korte2012} 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 \( k = \max_{S \in I} |S| \) 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 \( 1_S \) 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 \( P \) be the convex hull of \( \{1_S : S \in I\} \). Note that \( P \) 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 minimizing 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 \( x \in \mathbb{R}_+^n \), where \( x_i \) is the probability that element \( i \in X \) is chosen, a BRI oracle computes \( \arg \min_{f \in F} \mathbb{E}_{S \sim P} [f(S)] \). We use \( S \sim x \) to denote that \( S \) is drawn from the independent distribution with marginals \( x \). 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, Schult, 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; Bosansky et al., 2015; Halvorson, Conitzer, and Parr, 2009). Double oracle starts with each player restricted to only a small number of arbitrarily chosen 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. By contrast, our algorithm always respects the budget, but makes a randomized (instead of deterministic) decision. This randomized variant was studied by (Krause, Roper, and Golovin, 2011; Wilder et al., 2017), who gave algorithms for settings with small \(|\mathcal{F}|\). We need new techniques to handle exponentially large \(|\mathcal{F}|\).
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

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

Equivalently, $F(x) = \mathbb{E}_{S \sim x}[f(S)]$. 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 $\mathbb{E}[f(R)] \geq F(x)$. 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 outputting 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 $I$, 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 $\mathbb{E}_{S \sim p} \left[ \sum_{j \in S} w_j \right] =$
\[ \sum_{j=1}^n w_j \Pr[j \in S] = \sum_{j=1}^n w_j x_j, \] 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 \( p^x_\mu \)). The expected value of \( x \) 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 \( x \), \( G(x) \) is exactly the objective value of \( p^x_\mu \) 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 \( p_{\text{OPT}} \), then the independent distribution with the same marginals as \( p_{\text{OPT}} \) has at least \((1 - 1/e)\) of \( p_{\text{OPT}} \)'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 \( x \) to a distribution over the pure strategies \( I \). 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 \( \sum_j x_j \leq k \). 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 \( \Theta \left( \frac{1}{\delta} (\log |\mathcal{F}| + \log \frac{1}{\delta}) \right) \) independent samples via swap rounding and return the uniform distribution over the sampled pure strategies.

To sum up, our strategy is as follows. First, solve the continuous optimization problem to obtain marginal vector \( x \). Second, draw sampled pure strategies by running randomized swap rounding on \( x \).

### 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 \( G \) (proofs are given in the supplement). 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.

The proof is similar to the proof that the minimum of concave functions is concave. Up-concavity of \( G \) is the crucial property that enables efficient optimization.

Second, \( G \) is Lipschitz. Specifically, let \( M = \max_{j \in \mathcal{J}} f_j(\{j\}) \) 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 \( \nabla F_i(x) \). In general, \( \nabla F_i(x) \) 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 \( \max_{x \in \mathcal{P}} G(x) \). 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 \( G_\mu(x) = \sum_{j=1}^n w_j (x_j + \delta)^+ \), where \( (x_j + \delta)^+ = \max(x_j + \delta, 0) \). This smooths the objective and enables efficient optimization using standard algorithms.

---

(1) **Note:** The expected value of \( x \) under any submodular function is exactly given by its multilinear extension, which is a continuous function.
direction $v$ is in contrast to gradient-based algorithms for concave optimization, which move in the (possibly negative) direction of $v$. To carry out these steps, SFW requires three oracles. First, a linear optimization oracle $LO$ which, given an objective $u$, returns $\max_{v \in \mathcal{P}} u^\top v$. In the context of our problem, $LO$ outputs the indicator vector of the set $S \in \mathcal{I}$ which maximizes the linear objective $u$. $S$ 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 $x$, returns an unbiased estimate of $\nabla F_i(x)$.

The algorithm starts at $x^0 = 0$. At each iteration $\ell$, it averages over $m$ calls to $FO$ to compute a stochastic approximation $\nabla\ell$ to $\nabla G_{\mu}(x^{\ell-1})$ (Lines 4-9). For each call, it draws a random perturbation $z \sim \mu(u)$ and uses the BRI to find the minimizing $F_\ell$ at $x^{\ell-1} + z$. It then queries $FO$ for an estimate of $\nabla F_\ell(x^{\ell-1} + z)$. Lastly, it takes a step in the direction of $v^\ell = LO(\nabla\ell)$ by setting $x^\ell = x^{\ell-1} + \frac{1}{\ell_1} v^\ell$ (Lines 10-11). Since $x^\ell$ 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 $v^\ell$ is always nonnegative). This is in contrast to gradient-based algorithms for concave optimization, which move in the (possibly negative) direction $v^\ell - x^\ell$. An up-concave function, $G_\mu$ is concave along all nonnegative directions. By moving only

\[ \mathbb{E}_{z \sim \mu(u)} [G(x + z)] \] which averages over the region around $x$. This (and similar) techniques have been studied in the context of convex optimization \cite{duchi2012}. 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.
- $|G_\mu(x) - G(x)| \leq \frac{Muu}{2} \quad \forall x$.
- $G_\mu$ is differentiable, with $\nabla G_\mu(x) = \mathbb{E}[\nabla G(x + z)]$.
- $\nabla G_\mu$ is $\frac{M}{\ell_1}$-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 requires three oracles. First, a linear optimization oracle $LO$ which, given an objective $w$, returns argmax$_{v \in \mathcal{P}} w^\top v$. In the context of our problem, $LO$ outputs the indicator vector of the set $S \in \mathcal{I}$ which maximizes the linear objective $w$. $S$ 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 $F_\ell$ and point $x$, returns an unbiased estimate of $\nabla F_\ell(x)$.

The algorithm starts at $x^0 = 0$. At each iteration $\ell$, it averages over $m$ calls to $FO$ to compute a stochastic approximation $\nabla\ell$ to $\nabla G_{\mu}(x^{\ell-1})$ (Lines 4-9). For each call, it draws a random perturbation $z \sim \mu(u)$ and uses the BRI to find the minimizing $F_\ell$ at $x^{\ell-1} + z$. It then queries $FO$ for an estimate of $\nabla F_\ell(x^{\ell-1} + z)$. Lastly, it takes a step in the direction of $v^\ell = LO(\nabla\ell)$ by setting $x^\ell = x^{\ell-1} + \frac{1}{\ell_1} v^\ell$ (Lines 10-11). Since $x^\ell$ 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 $v^\ell$ is always nonnegative). This is in contrast to gradient-based algorithms for concave optimization, which move in the (possibly negative) direction $v^\ell - x^\ell$. As an up-concave function, $G_\mu$ is concave along all nonnegative directions. By moving only

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

1: $x^0 \leftarrow u1$
2: //Stochastic Frank-Wolfe algorithm
3: for $\ell = 1 \ldots K$ do
4:   for $t = 1 \ldots m$ do
5:     Draw $z \sim \mu(u)$
6:     $F_\ell \leftarrow BRI(x^{\ell-1} + z)$
7:     $\nabla\ell \leftarrow FO(F_\ell, x^{\ell-1} + z)$
8:   end for
9:   $\nabla\ell \leftarrow \frac{1}{m} \sum_{t=1}^{m} \nabla\ell$
10:  $v^\ell \leftarrow LO(\nabla\ell)$
11:  $x^\ell \leftarrow x^{\ell-1} + \frac{1}{\ell_1} v^\ell$
12: end for
13: $x_{final} \leftarrow x^K - u1$
14: //Sample from equilibrium mixed strategy
15: Return $r$ samples of SwapRound($x_{final}$)
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 \(z\) could result in negative values, for which the multilinear extension is not defined. To circumvent this, we start the algorithm at \(x^0 = u1\) (i.e., with small positive values in every entry) and then return \(x_{\text{final}} = x^K - u1\) (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 \(x^K\) satisfying 

\[
G(x^K) \geq (1 - \frac{1}{e})OPT - \epsilon \quad \text{with probability at least } 1 - \delta.
\]

Its runtime is \(\tilde{O} \left( \frac{TM^2k^2n}{\epsilon^2} + \frac{k^2M^4n}{\epsilon^3} \log \left( \frac{1}{\delta} \right) \right)^2\)

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 \(T_1 = \mathcal{O}(n \log n)\) 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 \(x_{\text{final}}\).

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

**Proof.** Suppose that \(p_{\text{OPT}}\) 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 \(\min_i \mathbb{E}_{S \sim p_{\text{OPT}}^i}[f_i(S)]\). With slight abuse of notation, let \(p_{\text{OPT}}^i\) be the independent distribution with the same marginals as \(p_{\text{OPT}}\). By applying the correlation gap to each \(f_i \in \mathcal{F}\) and taking the min, we have

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

By definition of \(x^*\), \(G(x^*) \geq \min_{f_i \in \mathcal{F}} \mathbb{E}_{S \sim p_{\text{OPT}}^i}[f_i(S)]\). Hence, \(G(x^*) \geq (1 - \frac{1}{e})^2OPT - \epsilon \geq (1 - \frac{1}{e})^2OPT - \epsilon\). 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.

This guarantee is sufficient if all 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 \(r = \mathcal{O} \left( \frac{1}{\delta} \left( \log |\mathcal{F}| + \log \frac{1}{\delta} \right) \right)\) 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 \(\mathcal{O} \left( \frac{eM^2k^2n}{\epsilon} \right)\).

**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).

\[^1\]The \(\tilde{O}\) notation hides logarithmic terms
Network security games: Earlier, we formulated network security games in the SBR framework. All we need to do is let EQUATOR be 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 \( \tau_i \). Let \( P(t_j) \) be the set of all paths from the (super)source 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 \( \min_i \mathbb{E}_{S \sim x} [f_i(S)] \). We can rewrite this as

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

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 a weight \( 1 \). Each customer \( v \) of the set of all paths from the (super)source to \( t_j \). Let \( f_i \) be the corresponding submodular objective. Given

\[a \mathbb{E}_{S \sim x} [f_i(S)]\]. We can rewrite this as

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

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 budget allocation:** Many widespread applications of submodular functions concern budget allocation, sensor placement, facility location, and many other common submodular optimization problems. Here we consider a robust budget allocation 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 \( v \) 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 \( s \) of \( R \). The advertiser solves the problem

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

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 \( s \) 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, Miyauchi et al. 2015).

In the robust optimization problem, the profits \( w \) 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 \( w \). 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 \( \hat{w} \) as

\[2\text{We use this formulation for simplicity, but it is possible to use only log B copies of each node (Ene and Nguyen 2016).} \]
\[ \mathcal{U}_{w}^w = \{ w : \exists c \in [0, 1]^{|R|} \text{ s.t. } w = (1 - c_i) \hat{w}_i, \ |c|_1 \leq \gamma \}. \]

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

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

Note that the inner expectation (which is the total probability that each \( v \in R \) is reached) is constant with respect to \( w \). Hence, the adversary’s best response problem of computing \( \min_{w \in \mathcal{U}_{w}^w} \mathbb{E}_{y \sim p}[f_w(y)] \) 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 \( \mathcal{U}_{w}^w \), 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, Conitzer, and Tambe 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 randomly drawn instances.

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 2(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 2(c) and 2(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 $\hat{w}$. 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 $[0, 0.2]$. $\hat{w}$ was randomly generated with each coordinate uniform in $[0.5, 1.5]$. Our uncertainty set is the D-norm set around $\hat{w}$ with $\gamma = \frac{1}{4}n$, 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 $K = 20, m = 10, \alpha = 0.1$.

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 $\mathcal{U}_\gamma(w)$, 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 the synthetic experiments. 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.

![Figure 2: Experimental results for budget allocation.](image-url)
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.; Ganzu, I.; and Tennenholtz, M. 2012. Optimizing budget allocation among channels and influencers. In WWW, 381–388. ACM.

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.

Bosanský, B.; Jiang, A. X.; Tambe, M.; and Kiekintveld, C. 2015. Combining compact representation and incremental generation in large games with sequential strategies. In AAAI.

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.

Delage, E.; Kuhn, D.; and Wiesemann, W. 2016. Decision 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.; Pêchouček, 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.; Vygen, J.; Korte, B.; and Vygen, J. 2012. Combinatorial optimization, volume 2. Springer.

Krause, A., and Golovin, D. 2014. Submodular function maximization.

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.
Appendix: Omitted proofs

We start out by proving some lemmas from the main text.

Proof of Lemma 1. Let \( u \geq 0 \). We would like to show that for any \( x \) and any \( \xi \geq 0 \), \( G(x + \xi u) \) is concave as a function of \( \xi \). Fix any \( \xi_1, \xi_2 \geq 0 \) and any \( \lambda \in [0, 1] \). We have

\[
\min_i F_i(x + (\lambda \xi_1 + (1 - \lambda)\xi_2)u) \geq \min_i [\lambda F_i(x + \xi_1 u) + (1 - \lambda)F_i(x + \xi_2 u)] \\
\geq \lambda \min_i F_i(x + \xi_1 u) + (1 - \lambda) \min_i F_i(x + \xi_2 u)
\]

where the first inequality follows because each \( F_i \) is individually up-concave.

Proof of Lemma 2. \( G \) is differentiable at a point \( x \) precisely when there is a unique \( F_i \) such that \( F_i(x) = \min_j F_j(x) \). Here, we have \( \nabla G(x) = \nabla F_i(x) \). Note that \( \frac{\partial F_i}{\partial x_j}(x) = \mathbb{E}[f_i(R(x) | j \in R(x))] - \mathbb{E}[f_i(R(x) | j \not\in R(x))] = \mathbb{E}[f_i(j | R(x - x_j))] \). By submodularity, we conclude that \( \frac{\partial F_i}{\partial x_j}(x) \leq f_i(\{j\}) \leq M \). Further, \( \frac{\partial G}{\partial x_j}(x) \geq 0 \) always holds by monotonicity. Thus, \( \|\nabla G(x)\|_{\infty} \leq M \).

Let \( \mu \) be the uniform probability distribution over the \( \ell_\infty \) ball of radius \( u \). Define the smoothed function \( G_\mu(x) = \mathbb{E}_{z \sim \mu}[G(x + z)] \). We will show the following properties of \( G_\mu \):

Proof of Lemma 3. For the first property, we start out by fixing the draw of \( z \) from \( \mu \). Following the logic of Lemma 1, we have that

\[
\min_i F_i(x + z + (\lambda \xi_1 + (1 - \lambda)\xi_2)u) \geq \min_i [\lambda F_i(x + z \xi_1 u) + (1 - \lambda)F_i(x + z + \xi_2 u)] \\
\geq \lambda \min_i F_i(x + z + \xi_1 u) + (1 - \lambda) \min_i F_i(x + z + \xi_2 u).
\]
Since these inequalities hold for any fixed $z$, they also hold in expectation over a random $z$, so we conclude that $G_\mu$ is up-concave.

For the second property: since $||\nabla G||_\infty \leq M$, $G$ is $M$-Lipschitz with respect to the $\ell_1$ norm. Thus, we have
\[
E[G(x + z)] \leq G(x) + M \mathbb{E}[||z||_1] \leq G(x) + \frac{Mnu}{2}
\]
and analogously, $E[G(x + z)] \geq G(x) - \frac{Mnu}{2}$.

The third property follows from the fact that $G$ is differentiable almost everywhere. To see this, note that $G$ is differentiable wherever there is a unique minimizing $F_i$, in which case $\nabla G = \nabla F_i$. Suppose that there is not a unique minimizer at some point $x$. There are two cases. First, if there is an open ball around $x$ such that the minimizing functions at $x$ coincide at every point in the ball, then their gradients also coincide in the ball. Thus, $G$ is still differentiable at $x$. Second, if no such open ball exists, then the set of points at which $G$ is not differentiable has measure zero. Hence, taking a random perturbation of the input avoids such points with probability 1.

For the proof of the fourth property, we follow the argument of Duchi et al. (2012). We first claim that
\[
||\nabla G_\mu(x) - \nabla G_\mu(y)||_\infty = ||E[\nabla G(x + z)] - E[\nabla G(y + z)]||_\infty \leq M \int |\mu(z - x) - \mu(z - y)|dz.
\]  

(2)

We prove this claim as follows. Without loss of generality, we take $x = 0$ for this step of the proof (via a linear change of variables). Let $g(x)$ be a function that is defined as $\nabla G(x)$ where $G$ is differentiable. At the (measure 0) set of points where $G$ is not differentiable, we define $g$ to be equal to $\nabla F_i(x)$ for an arbitrary $i \in \arg \min_j F_j(x)$. With probability 1, $E[g(x + z)] = E[\nabla G(x + z)]$. We have
\[
E[g(z) - g(y + z)] = \int g(z)\mu(z)dz - \int g(y + z)\mu(z)dz
\]
\[
= \int g(z)\mu(z)dz - \int g(y)\mu(z - y)dz
\]
\[
= \int g(z)\mu(z - y)dz + \int g(z)\mu(z - y)dz
\]
where $I_\geq = \{z | \mu(z) > \mu(z - y)\}$ and $I_\leq = \{z | \mu(z) < \mu(z - y)\}$. Taking norms, we have
\[
||E[g(z) - g(y + z)]||_\infty \leq \sup_{z \in I_\geq \cup I_\leq} ||g(z)||_\infty \int_{I_\geq} [\mu(z) - \mu(z - y)] + \int_{I_\leq} [\mu(z - y) - \mu(z)]
\]
\[
\leq M \int_{I_\geq} [\mu(z) - \mu(z - y)] + \int_{I_\leq} [\mu(z - y) - \mu(z)]
\]
\[
= M \int |\mu(z) - \mu(z - y)|dz
\]

Having proved that Equation (2) holds, we now just need to show $\int |\mu(z - x) - \mu(z - y)|dz \leq \frac{||x - y||_1}{u}$. This follows from Duchi et al. (2012), Lemma 12.

We now prove a technical smoothness lemma. The argument is standard, but we include it for completeness.

**Lemma 4.** For any $x, y$, $G_\mu(x + \gamma y) - G_\mu(x) \geq \gamma \nabla G_\mu(x)^T y - \frac{Mk^2\gamma^2}{2u}$.  

**Proof.** For any $x, y \in \mathcal{P}$, we consider the one dimensional auxiliary function $g_{x,y}(\xi) = G_\mu(x + \xi y)$. We have
Lemma 5. For any DR-submodular function $G$ and its optimizer $x^*$, $G(x^* + x) - G(x) \leq \nabla G(x)^\top x^*$. We can now proceed to prove our guarantee on the performance of the SFW algorithm for optimizing the objective $G$.

Proof of Theorem[7] We analyze the gain made in a single step of SFW as follows:

\[
G_\mu(x^\ell) - G_\mu(x^{\ell-1}) \geq \gamma_\ell \nabla G_\mu(x^{\ell-1})^\top v^\ell - \frac{Mk^2}{2u} \gamma_\ell^2 \text{ (Lemma[4])}
\]

\[
= \gamma_\ell \tilde{v}^\top v^\ell - \gamma_\ell \left( \tilde{\nabla} \ell - \nabla G_\mu(x^{\ell-1}) \right)^\top v^\ell - \frac{Mk^2}{2u} \gamma_\ell^2
\]

\[
\geq \gamma_\ell \tilde{v}^\top v^\ell - \gamma_\ell k \|\tilde{\nabla} \ell - \nabla G_\mu(x^{\ell-1})\|_\infty - \frac{Mk^2}{2u} \gamma_\ell^2 \text{ (by Holder’s inequality and rank($\mathcal{M}$) = k)}
\]

\[
\geq \gamma_\ell \tilde{\nabla}^\top \ell x^* - \gamma_\ell k \|\tilde{\nabla} \ell - \nabla G_\mu(x^{\ell-1})\|_\infty - \frac{Mk^2}{2u} \gamma_\ell^2 \text{ (by definition of } x^*)
\]

\[
\geq \gamma_\ell \nabla G_\mu(x^{\ell-1})^\top x^* - \gamma_\ell \left( G_\mu(x^{\ell-1}) - \tilde{\nabla} \ell \right)^\top \ell x^* - \gamma_\ell k \|\tilde{\nabla} \ell - \nabla G_\mu(x^{\ell-1})\|_\infty - \frac{Mk^2}{2u} \gamma_\ell^2
\]

\[
\geq \gamma_\ell \nabla G_\mu(x^{\ell-1})^\top x^* - 2\gamma_\ell k \|\tilde{\nabla} \ell - \nabla G_\mu(x^{\ell-1})\|_\infty - \frac{Mk^2}{2u} \gamma_\ell^2
\]

\[
\geq \gamma_\ell (G_\mu(x^* + x^{\ell-1}) - G_\mu(x^{\ell-1})) - 2\gamma_\ell k \|\tilde{\nabla} \ell - \nabla G_\mu(x^{\ell-1})\|_\infty - \frac{Mk^2}{2u} \gamma_\ell^2 \text{ ( Lemma[5] and } x^* \geq 0)
\]

\[
\geq \gamma_\ell (G_\mu(x^*) - G_\mu(x^{\ell-1})) - 2\gamma_\ell k \|\tilde{\nabla} \ell - \nabla G_\mu(x^{\ell-1})\|_\infty - \frac{Mk^2}{2u} \gamma_\ell^2 \text{ (monotonicity)}
\]

Now we give a high probability bound on $\|\tilde{\nabla} \ell - \nabla G_\mu(x^{\ell-1})\|_\infty$. Denote by $\tilde{\nabla}^i_\ell$ the $i$th randomly sampled gradient and $\left[\tilde{\nabla}^j_\ell\right]$ its $j$th entry (the derivative with respect to item $j$). We will give a high probability bound on each individual entry of the estimated gradient and then combine them using union bound to control $\|\tilde{\nabla} \ell - \nabla G_\mu(x^{\ell-1})\|_\infty$. 

\[
G_\mu(x + \gamma y) - G_\mu(x) = \int_{\xi=0}^{1} \frac{dG_\mu(x + y(\xi))}{d\xi} d\xi
\]

\[
= \int_{\xi=0}^{1} \nabla G_\mu(x + \xi y)^\top (\gamma y) d\xi
\]

\[
= \gamma \int_{\xi=0}^{1} \nabla G_\mu(x)^\top y + \left[\nabla G_\mu(x + \xi y)^\top - \nabla G_\mu(x)^\top \right] y d\xi
\]

\[
\geq \gamma \int_{\xi=0}^{1} \nabla G_\mu(x)^\top y - \|\nabla G_\mu(x + \xi y)^\top - \nabla G_\mu(x)^\top\|_\infty \|y\|_1 d\xi \text{ (by Holder’s inequality)}
\]

\[
\geq \gamma \int_{\xi=0}^{1} \nabla G_\mu(x)^\top y - \frac{M}{\mu} \|\xi y\|_1 \|y\|_1 d\xi \text{ (by Lipschitz)}
\]

\[
\geq \gamma \nabla G_\mu(x)^\top - \gamma^2 \int_{\xi=0}^{1} \frac{MK^2}{u} \xi d\xi (\text{bound on } \ell_1 \text{ diameter of } \mathcal{P})
\]

which proves the lemma. \[\square\]
Fix any individual entry \( j \). We have \( \sum_{i=1}^{m} [\hat{\nabla}_i]_j = \frac{1}{m} \sum_{i=1}^{m} [\hat{\nabla}_i]_j - \nabla_j G_\mu(x^{\ell-1}) \). Because the first-order oracle returns an unbiased estimate, we know that \( \mathbb{E} \left[ [\hat{\nabla}_i]_j - \nabla_j G_\mu(x^{\ell-1}) \right] = 0 \). Further, \( \left| [\hat{\nabla}_i]_j \right| \leq M \) and \( \left| \nabla_j G_\mu(x^{\ell-1}) \right| \leq M \), so \( \left| [\hat{\nabla}_i]_j - \nabla_j G_\mu(x^{\ell-1}) \right| \leq 2M \) holds via triangle inequality. Now via Hoeffding’s inequality, we have that

\[
\Pr \left[ \sum_{i=1}^{m} [\hat{\nabla}_i]_j - m \nabla_j G_\mu(x^{\ell-1}) \geq m \frac{\epsilon}{8k} \right] \leq 2e^{-\frac{m^2}{128M^2}}
\]

and so taking \( m = \frac{128M^2k^2 \log \frac{4Kn}{\delta}}{\epsilon^2} \) ensures that

\[
\Pr \left[ \left| [\hat{\nabla}_i]_j - \nabla_j G_\mu(x^{\ell-1}) \right| \geq \frac{\epsilon}{8k} \right] \leq \frac{\delta}{2Kn}.
\]

By union bound, the total probability of this event holding for all \( n \) items at each of the \( K \) timesteps is at least \( 1 - \frac{\delta}{2} \). In all of what follows, we condition on this happening. Rearranging gives

\[
G_\mu(x^*) - G_\mu(x^K) \leq (1 - \gamma_\ell) \left[ G_\mu(x^*) - G_\mu(x^{\ell-1}) \right] - 2\gamma_\ell \epsilon \| [\nabla_\ell - \nabla_\ell G_\mu(x^{\ell-1})] \|_\infty - \frac{MK^2}{2u} \gamma_\ell^2
\]

and so after \( K \) iterations we obtain

\[
G_\mu(x^*) - G_\mu(x^K) \leq \prod_{\ell=0}^{K-1} (1 - \gamma_\ell) \left[ G_\mu(x^*) - G_\mu(x^0) \right] - \sum_{\ell=0}^{K-1} \frac{\gamma_\ell \epsilon}{4} - \sum_{\ell=0}^{K-1} \frac{MK^2}{2u} \gamma_\ell^2
\]

with constant stepsize \( \gamma = \frac{1}{K} \), we have \( \sum_{\ell=0}^{K-1} \gamma_\ell = 1 \) and \( \sum_{\ell=0}^{K-1} \gamma_\ell^2 = \frac{1}{K} \), this yields

\[
G_\mu(x^*) - G_\mu(x^K) \leq \frac{1}{e} G_\mu(x^*) - \frac{\epsilon}{4} - \frac{MK^2}{2uK}
\]

and hence

\[
G(x^*) - G(x^K) \leq \frac{1}{e} G_\mu(x^*) - \frac{\epsilon}{4} - \frac{MK^2}{2uK} - Mnu.
\]

and taking \( u = \frac{\epsilon}{4Mn} \) and \( K = \frac{8M^2k^2n \log \frac{4Kn}{\delta}}{\epsilon^2} \) ensures that \( G(x^*) - G(x^K) \leq \frac{1}{e} G_\mu(x^*) - \frac{3\epsilon}{4} \).

Now we just need a small trick to deal with the issue that \( G \) is only defined for \( x \in [0, 1]^n \), and random perturbation by \( z \) may take us out of this range. To avoid negative values, we start the algorithm at the point \( u1 \). Since each coordinate only increases during the course of the algorithm, we are guaranteed to query \( G \) only at nonnegative points. To deal with values greater than 1, we can instead analyze the function \( H(x) = G(x \land 1) \), where \( \land \) denotes coordinate-wise maximum. \( H \) is also up-concave, and agrees with \( G \) at every point in \( \mathcal{P} \). After running SFW applied to \( H \) for \( K \) iterations, we obtain via Theorem 1 a solution \( x^K \) such that \( H(x^K) \geq (1 - \frac{1}{2}) \max_{x \in \mathcal{P}} H(x) - \epsilon = (1 - \frac{1}{2}) \max_{x \in \mathcal{P}} G(x) - \epsilon \). The issue is that we may have \( x^K \not\in \mathcal{P} \). We output the point \( x^K - u1 \), which is guaranteed to lie in \( \mathcal{P} \). To analyze the loss incurred, we use the following lemma

15
Lemma 6. Let \( f \) be a monotone submodular function with \( \max_i f(\{i\}) \leq M \). Let \( R(x) \) be a random set in which every element appears independently with probability \( x_i \geq u \). Then \( \mathbb{E}[f(R(x) - u1)] \geq \mathbb{E}[f(R(x))] - uMn \).

Proof. We decompose the expected value of \( R(x) - u1 \) into the expected marginal contribution from each item:

\[
\mathbb{E}[f(R(x) - u1)] = \sum_{i=1}^{n} \Pr[i \in f(R(x) - u1)] \mathbb{E}[f(i)(R(x) - u1)] \quad \text{(linearity of expectation)}
\]

\[
\geq \sum_{i=1}^{n} \Pr[i \in f(R(x) - u1)] \mathbb{E}[f(i)(R(x))]
\]

\[
\geq \sum_{i=1}^{n} (x_i - u) \mathbb{E}[f(i)(R(x))]
\]

\[
= \sum_{i=1}^{n} x_i \mathbb{E}[f(i)(R(x))] - u \sum_{i=1}^{n} \mathbb{E}[f(i)(R(x))]
\]

\[
\geq \sum_{i=1}^{n} x_i \mathbb{E}[f(i)(R(x))] - u \mathbb{E}[f(\{i\})] \quad \text{(submodularity)}
\]

\[
\geq \sum_{i=1}^{n} x_i \mathbb{E}[f(i)(R(x))] - uMn \quad \text{\( (f(\{i\}) \leq M) \)}
\]

\[
= \mathbb{E}[f(R(x)))] - uMn.
\]

Applying Lemma 6 to every \( f_i \in F \), we conclude that

\[
H(x^K - u1) \geq H(x^K) - uMn = G(x^K) - \frac{e}{4} \geq \left(1 - \frac{1}{e}\right) G(x^*) - \epsilon
\]

which completes the proof.

Lastly, we prove our concentration guarantee for the output of the swap rounding algorithm.

Proof of Theorem 3. For now, fix a specific function \( f_i \). We will show that with high probability, the expected value of \( f_i \) on the empirical distribution is close to its expected value on the full distribution induced by randomized swap rounding. At the end we will take a union bound over all \( f_i \in F \). Let \( R_\ell \) be the random set drawn in the \( \ell \)th iteration of randomized swap rounding. Let \( \mu_0 = F_\ell(x^K) \). Note that for all \( \ell \), \( \mathbb{E}[f_i(R_\ell)] \geq \mu_0 \) via the guarantee for randomized swap rounding, Let \( Y = \sum_{\ell=1}^{r} f_i(R_\ell) \) and note that \( \mathbb{E}[Y] \geq r\mu_0 \).

Our high-level approach is to apply Markov’s inequality to the random variable \( e^{\lambda \mu_0 - Y} \). Let \( \lambda \) be an arbitrary parameter in \([0, 1]\) (later, we will set \( \lambda \) to get the best bound). For any single iteration of randomized swap rounding, Chekuri et al. bound the exponential moment of the random variable \( \lambda (\mu_0 - f_i(R_\ell)) \) as

\[
\mathbb{E}[e^{\lambda(\mu_0 - f_i(R_\ell))}] \leq e^{2\lambda^2 \mu_0}.
\]

Since \( r\mu_0 - Y = \sum_{\ell=1}^{r} \mu_0 - f_i(R_\ell) \), we have

\[
\mathbb{E}\left[e^{\lambda(r\mu_0 - Y)}\right] = \mathbb{E}\left[e^{\sum_{\ell=1}^{r} \lambda(\mu_0 - f_i(R_\ell))}\right] = \mathbb{E}\left[\prod_{\ell=1}^{r} e^{\lambda(\mu_0 - f_i(R_\ell))}\right] = \prod_{\ell=1}^{r} \mathbb{E}\left[e^{\lambda(\mu_0 - f_i(R_\ell))}\right] \quad \text{(independence)}
\]

\[
\leq e^{2r\lambda^2 \mu_0}.
\]
Applying Markov’s inequality yields

\[
\Pr[r\mu_0 - Y \geq \epsilon r \mu_0] = \Pr\left[e^{\lambda(r\mu_0 - Y)} \geq e^{\epsilon r \lambda \mu_0}\right] \\
\leq \frac{\mathbb{E}\left[e^{\lambda(r\mu_0 - Y)}\right]}{e^{\epsilon r \lambda \mu_0}} \\
\leq e^{2r\lambda^2 \mu_0 - \epsilon r \lambda \mu_0}
\]

Taking \( \lambda = \frac{\epsilon}{4} \), we obtain

\[
\Pr\left[\frac{1}{r} \sum_{\ell=1}^{r} f_i(R_\ell) \leq (1 - \epsilon) \mu_0\right] = \Pr[r\mu_0 - Y \geq \epsilon r \mu_0] \leq e^{-r\mu_0^2/8}.
\]

We now distinguish two cases. First, \( \mu_0 < \epsilon \). Since \( f_i(R_\ell) \geq 0 \) \( \forall i, \ell, \frac{1}{r} \sum_{\ell=1}^{r} f_i(R_\ell) \geq \mu_0 - \epsilon \) holds with probability 1. Second, \( \mu_0 \geq \epsilon \). Here, we see that setting \( r = \Theta\left(\frac{1}{\epsilon} \left(\log |F| + \log \frac{1}{\delta}\right)\right) \) ensures that \( \frac{1}{r} \sum_{\ell=1}^{r} f_i(R_\ell) \geq (1 - \epsilon) \mu_0 \) holds with probability at least \( 1 - \frac{\delta}{|F|} \). Taking union bound over all \( f_i \in F \) completes the proof.

\( \square \)