Optimal Algorithms for Continuous Non-monotone Submodular and DR-Submodular Maximization

Rad Niazadeh
Department of Computer Science
Stanford University, Stanford, CA 95130
rad@cs.stanford.edu

Tim Roughgarden†
Department of Computer Science
Stanford University, Stanford, CA 95130
tim@cs.stanford.edu

Joshua R. Wang
Google, Mountain View, CA 94043
joshuawang@google.com

Abstract

In this paper we study the fundamental problems of maximizing a continuous non-monotone submodular function over a hypercube, with and without coordinate-wise concavity. This family of optimization problems has several applications in machine learning, economics, and communication systems. Our main result is the first \(\frac{1}{2}\)-approximation algorithm for continuous submodular function maximization; the approximation factor of \(\frac{1}{2}\) is the best possible for algorithms that use only polynomially many queries. For the special case of DR-submodular maximization, i.e., when the submodular functions is also coordinate-wise concave along all coordinates, we provide a faster \(\frac{1}{2}\)-approximation algorithm that runs in almost linear time. Both of these results improve upon prior work [Bian et al., 2017a, b, Soma and Yoshida, 2017, Buchbinder et al., 2012, 2015].

Our first algorithm is a single-pass algorithm that uses novel ideas such as reducing the guaranteed approximation problem to analyzing a zero-sum game for each coordinate, and incorporates the geometry of this zero-sum game to fix the value at this coordinate. Our second algorithm is a faster single-pass algorithm that exploits coordinate-wise concavity to identify a monotone equilibrium condition sufficient for getting the required approximation guarantee, and hunts for the equilibrium point using binary search. We further run experiments to verify the performance of our proposed algorithms in related machine learning applications.

1 Introduction

Submodular optimization is a sweet spot between tractability and expressiveness, with numerous applications in machine learning (e.g., Krause and Golovin [2014], and see below) and with many algorithms that are both practical and enjoy good rigorous guarantees (e.g., Buchbinder et al. [2012, 2015]). In general, a real-valued function \(F\) defined on a lattice \(L\) is submodular if and only if

\[
F(x \lor y) + F(x \land y) \leq F(x) + F(y)
\]

for all \(x, y \in L\), where \(x \lor y\) and \(x \land y\) denote the join and meet, respectively, of \(x\) and \(y\) in the lattice \(L\). Such functions are generally neither convex nor concave. In one of the most commonly studied examples, \(L\) is the lattice of subsets of a fixed ground set (or a sublattice thereof), with union and intersection playing the roles of join and meet, respectively.

*Rad Niazadeh was supported by Stanford Computer Science Motwani Fellowship.
†Tim Roughgarden was supported in part by Google Faculty Grant, Guggenheim Fellowship, and NSF Grants CCF-1524062 and CCF-1813188.
This paper concerns a different well-studied setting, where $L$ is a hypercube (i.e., $[0, 1]^n$), with componentwise maximum and minimum serving as the join and meet, respectively.\footnote{Our results also extend easily to arbitrary axis-aligned boxes (i.e., “box constraints”).} We consider the fundamental problem of (approximately) maximizing a continuous and nonnegative submodular function over the hypercube.\footnote{More generally, the function only has to be nonnegative at the points $0$ and $\bar{1}$.} The function $F$ is given as a “black box” and can only be accessed by querying its value at a point. We are interested in algorithms that use at most a polynomial (in $n$) number of queries. We do not assume that $F$ is monotone (otherwise the problem is trivial).

We next briefly mention four applications of maximizing a non-monotone submodular function over a hypercube that are germane to machine learning and other related application domains.\footnote{See the supplement for more details on these applications.}

Non-concave quadratic programming. In this problem, the goal is to maximize $F(x) = \frac{1}{2}x^T H x + h^T x + c$, where the off-diagonal entries of $H$ are non-positive. One application of this problem is to large-scale price optimization on the basis of demand forecasting models [Ito and Fujimaki, 2016].

Map inference for Determinantal Point Processes (DPP). DPPs are elegant probabilistic models that arise in statistical physics and random matrix theory. DPPs can be used as generative models in applications such as text summarization, human pose estimation, and news threading tasks [Kulesza et al., 2012]. The approach in Gillenwater et al. [2012] to the problem boils down to maximizing a suitable submodular function over the hypercube, accompanied with an appropriate rounding (see also [Bian et al., 2017a]). One can also think of regularizing this objective function with $\ell_2$-norm regularizer, to avoid overfitting, and the function will still remain submodular.

Log-submodularity and mean-field inference. Another probabilistic model that generalizes DPPs and all other strong Rayleigh measures [Li et al., 2016, Zhang et al., 2015] is the class of log-submodular distributions over sets, i.e., $p(S) \sim \exp(\mathcal{F}(S))$ where $\mathcal{F}(\cdot)$ is a set submodular function. MAP inference over this distribution has applications in machine learning [Djolonga and Krause, 2014]. One variational approach towards this MAP inference task is to use mean-field inference to approximate the distribution $p$ with a product distribution $x \in [0, 1]^n$, which again boils down to submodular function maximization over the hypercube (see [Bian et al., 2017a]).

Revenue maximization over social networks. Here, there is a seller who wants to sell a product over a social network of buyers. To do so, it freely assigns trial products and fractions thereof to the buyers in the network [Bian et al., 2017b, Hartline et al., 2008]. For this problem, one can reduce it to maximizing an objective function that takes into account two parts: the revenue gain from those who did not get a free product, where the revenue function for any such buyer is a non-negative non-decreasing and submodular function $R_0(x)$; and the revenue loss from those who received the free product, where the revenue function for any such buyer is a non-positive non-increasing and submodular function $R_1(x)$. The combination for all buyers is a non-monotone submodular function. It also is non-negative at $0$ and $\bar{1}$, by extending the model and accounting for extra revenue gains from buyers with free trials.

Our Results. Maximizing a submodular function over the hypercube is at least as difficult as over the subsets of a ground set.\footnote{An instance of the latter problem can be converted to one of the former by extending the given set function $f$ (with domain viewed as $\{0, 1\}^n$) to its multilinear extension $\mathcal{F}$ defined on the hypercube (where $\mathcal{F}(x) = \sum_{S \subseteq [n]} \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i) / f(S)$). Sampling based on an $\alpha$-approximate solution for the multilinear extension yields an equally good approximate solution to the original problem.} For the latter problem, the best approximation ratio achievable by an algorithm making a polynomial number of queries is $\frac{1}{2}$; the (information-theoretic) lower bound is due to [Feige et al., 2007, 2011], the optimal algorithm to [Buchbinder et al., 2012, 2015]. Thus, the best-case scenario for maximizing a submodular function over the hypercube (using polynomially many queries) is a $\frac{1}{2}$-approximation. The main result of this paper achieves this best-case scenario:

There is an algorithm for maximizing a continuous submodular function over the hypercube that guarantees a $\frac{1}{2}$-approximation while using only a polynomial number of queries to the function under mild continuity assumptions.

Our algorithm is inspired by the bi-greedy algorithm of Buchbinder et al. [2015], which maximizes a submodular set function; it maintains two solutions initialized at $0$ and $\bar{1}$, goes over coordinates...
We also run experiments to verify the performance of our proposed algorithms in practical machine learning applications. As simple corollaries, a twice-differentiable function can be submodular, are concave in each coordinate. This class of functions is called DR-submodular in the literature (e.g., in [Soma and Yoshida, 2015] and based on diminishing returns defined in [Kapralov et al., 2013]). Here, an optimal 1/4-approximation algorithm was recently already known on integer lattices [Soma and Yoshida, 2017], that can easily be generalized to our continuous setting as well; our contribution is a significantly faster such bi-greedy algorithm. The main idea here is to identify a monotone equilibrium condition sufficient for getting the required approximation guarantee, which enables a binary search-type solution. See Section 2 for more details.

The second and third applications above induce objective functions that, in addition to being submodular, are concave in each coordinate. This class of functions is called DR-submodular in the literature (e.g., in [Soma and Yoshida, 2015] and based on diminishing returns defined in [Kapralov et al., 2013]). Here, an optimal 1/4-approximation algorithm was recently already known on integer lattices [Soma and Yoshida, 2017], that can easily be generalized to our continuous setting as well; our contribution is a significantly faster such bi-greedy algorithm. The main idea here is to identify a monotone equilibrium condition sufficient for getting the required approximation guarantee, which enables a binary search-type solution. See Section 2 for more details.

We also run experiments to verify the performance of our proposed algorithms in practical machine learning applications. We observe that our algorithms match the performance of the prior work, while providing either a better guaranteed approximation or a better running time.

**Further Related Work.** Buchbinder and Feldman [2016] derandomize the bi-greedy algorithm. Staib and Jegelka [2017] apply continuous submodular optimization to budget allocation, and develop a new submodular optimization algorithm to this end. Hassani et al. [2017] give a 1/4-approximation for monotone continuous DR-submodular functions under convex constraints, which is later improved to (1 - ¼)-approximation in Mokhtari et al. [2018] (even for stochastic functions). Gotovos et al. [2015] consider (adaptive) submodular maximization when feedback is given after an element is chosen. Chen et al. [2018], Roughgarden and Wang [2018] consider submodular maximization in the context of online no-regret learning. Mirzasoleiman et al. [2013] show how to perform submodular maximization with distributed computation. Submodular minimization is studied in Schrijver [2000], Iwata et al. [2001]. See Bach et al. [2013] for a survey on more applications in machine learning.

**Variations of Continuous Submodularity.** We consider non-monotone non-negative continuous submodular functions, i.e., \( F : [0, 1]^n \rightarrow [0, 1] \) s.t. \( \forall x, y \in [0, 1]^n, F(x) + F(y) \geq F(x \lor y) + F(x \land y) \), where \( \lor \) and \( \land \) are coordinate-wise max and min operations. Two related properties are weak Diminishing Returns Submodularity (weak DR-SM) and strong Diminishing Returns Submodularity (strong DR-SM) [Bian et al., 2017b], formally defined below. Indeed, weak DR-SM is equivalent to submodularity (see Proposition 3 in the supplement), and hence we use these terms interchangeably.

**Definition 1 (Weak/Strong DR-SM).** Consider a continuous function \( F : [0, 1]^n \rightarrow [0, 1] \):

- **Weak DR-SM (continuous submodular):** \( \forall i \in [n], \forall x_{-i} \leq y_{-i} \in [0, 1]^n, \forall \delta \geq 0, \forall z \)
  \[
  F(z + \delta, x_{-i}) - F(z, x_{-i}) \geq F(z, y_{-i}) - F(z, y_{-i})
  \]

- **Strong DR-SM (DR-submodular):** \( \forall i \in [n], \forall x \leq y \in [0, 1]^n, \forall \delta \geq 0 \):
  \[
  F(x_i + \delta, x_{-i}) - F(x) \geq F(y_i + \delta, y_{-i}) - F(y)
  \]

As simple corollaries, a twice-differentiable \( F \) is strong DR-SM if and only if all the entries of its Hessian are non-positive, and weak DR-SM if and only if all of the off-diagonal entries of its Hessian are non-positive. Also, weak DR-SM together with concavity along each coordinate is equivalent to strong DR-SM (see Proposition 3 in the supplementary materials for more details).

**Coordinate-wise Lipschitz Continuity.** Consider univariate functions generated by fixing all but one of the coordinates of the original function \( F(\cdot) \). In future sections, we sometimes require mild technical assumptions on the Lipschitz continuity of these single dimensional functions.

---

\(^7\)However, after regularization the function still remains submodular, but can lose coordinate-wise concavity.
We start by describing a vanilla version of our algorithm for maximizing Theorem 1.

The Lipschitz continuous variate function implicitly require such an assumption to perform single-dimensional optimization. Prior works, e.g., Bian et al. (2017b) and Bian et al. (2017a), implicitly require such an assumption to perform single-dimensional optimization.

Definition 2 (Coordinate-wise Lipschitz). A function $\mathcal{F} : [0,1]^n \to [0,1]$ is coordinate-wise Lipschitz continuous if there exists a constant $C > 0$ such that $\forall i \in [n], \forall x_{-i} \in [0,1]^n$, the single variate function $\mathcal{F}(\cdot, x_{-i})$ is $C$-Lipschitz continuous, i.e.,

$$\forall z_1, z_2 \in [0,1] : \left| \mathcal{F}(z_1, x_{-i}) - \mathcal{F}(z_2, x_{-i}) \right| \leq C|z_1 - z_2|$$

2 Weak DR-SM Maximization: Continuous Randomized Bi-Greedy

Our first main result is a 2 Weak DR-SM Maximization: Continuous Randomized Bi-Greedy 2007.

2.2.2 We show how to approximately implement these oracles in polynomial time when $\mathcal{F}$ is coordinate-wise Lipschitz.

Theorem 1. If $\mathcal{F}(\cdot)$ is non-negative and continuous submodular (or equivalently is weak DR-SM), then Algorithm 1 is a randomized $\frac{1}{2}$-approximation algorithm, i.e., returns $\hat{z} \in [0,1]^n$ s.t.

$$2E [\mathcal{F}(\hat{z})] \geq \mathcal{F}(x^*), \quad \text{where } x^* \in \text{argmax } \mathcal{F}(x) \text{ is the optimal solution.}$$

\(\hat{z}\) is the optimal solution.
2.1 Analysis of the Continuous Randomized Bi-Greedy (Proof of Theorem 1)

We start by defining these vectors, used in our analysis in the same spirit as Buchbinder et al. [2015]:

\[ i \in [n] : X^{(i)}(\tilde{z}_i, 0, 0, \ldots, 0) , \quad Y^{(i)}(\tilde{z}_i, 1, 1, \ldots, 1) , \quad O^{(i)}(\tilde{z}_i, x^*_i, \ldots, x^*_n) \]

Note that \( X^{(i)} \) and \( Y^{(i)} \) (or \( X^{(i-1)} \) and \( Y^{(i-1)} \)) are the values of \( X \) and \( Y \) at the end of (or at the beginning of) the \( i \)th iteration of Algorithm 1. In the remainder of this section, we give the high-level proof ideas and present some proof sketches. See the supplementary materials for the formal proofs.

Reduction to Coordinate-wise Zero-sum Games. For each coordinate \( i \in [n] \), we consider a sub-problem. In particular, define a two-player zero-sum game played between the algorithm player (denoted by ALG) and the adversary player (denoted by ADV). ALG selects a (randomized) strategy \( \hat{z}_i \in [0, 1] \), and ADV selects a (randomized) strategy \( x^*_i \in [0, 1] \). Recall the descriptions of \( g(z) \) and \( h(z) \) at iteration \( i \) of Algorithm 1:

\[ g(z) = \mathcal{F}(z, X_{i}^{[i-1]}) - \mathcal{F}(Z, X_{i}^{[i-1]}) , \quad h(z) = \mathcal{F}(z, Y_{i}^{[i-1]}) - \mathcal{F}(Z, Y_{i}^{[i-1]}) \]

We now define the utility of ALG (negative of the utility of ADV) in our zero-sum game as follows:

\[ \mathcal{V}^{(i)}(\tilde{z}_i, x^*_i) = \frac{1}{2} g(\tilde{z}_i) + \frac{1}{2} h(\tilde{z}_i) - \max (g(x^*_i) - g(\tilde{z}_i), h(x^*_i) - h(\tilde{z}_i)) . \]

Suppose the expected utility of ALG is non-negative at the equilibrium of this game. In particular, suppose ALG’s randomized strategy \( \tilde{z}_i \) (in Algorithm 1) guarantees that for every strategy \( x^*_i \) of ADV the expected utility of ALG is non-negative. If this statement holds for all of the zero-sum games corresponding to different iterations \( i \in [n] \), then Algorithm 1 is a \( \frac{1}{2} \)-approximation of the optimum.

Lemma 1. If \( \forall i \in [n] : E[\mathcal{V}^{(i)}(\tilde{z}_i, x^*_i)] \geq -\delta/n \) for constant \( \delta > 0 \), then \( 2E[\mathcal{F}(\tilde{z})] \geq \mathcal{F}(x^*) - \delta \).
Algorithm 1: (Vanilla) Continuous Randomized Bi-Greedy

\textbf{input:} function $\mathcal{F} : [0, 1]^n \rightarrow [0, 1]$; \\
\textbf{output:} vector $\hat{z} = (\hat{z}_1, \ldots, \hat{z}_n) \in [0, 1]^n$; \\
Initialize $X \leftarrow (0, \ldots, 0)$ and $Y \leftarrow (1, \ldots, 1)$; \\
\textbf{for} $i = 1$ \textbf{to} $n$ \textbf{do} \\
\quad \text{Find } Z_u, Z_l \in [0, 1] \text{ such that} \begin{cases} 
Z_l \in \text{argmax}_{z \in [0, 1]} \mathcal{F}(z, X_{-i}) \\
Z_u \in \text{argmax}_{z \in [0, 1]} \mathcal{F}(z, Y_{-i}) 
\end{cases}; \\
\quad \text{if } Z_u \leq Z_l \text{ then} \\
\quad \quad \hat{z}_i \leftarrow Z_l; \\
\quad \text{else} \\
\quad \quad \forall z \in [Z_l, Z_u], \text{let} \begin{cases} 
g(z) \triangleq \mathcal{F}(z, X_{-i}) - \mathcal{F}(Z_l, X_{-i}), \\
h(z) \triangleq \mathcal{F}(z, Y_{-i}) - \mathcal{F}(Z_u, Y_{-i}), 
\end{cases}; \\
\quad \quad \text{Let } \alpha \triangleq g(Z_u) \text{ and } \beta \triangleq h(Z_l); \\
\quad \quad \text{// note that } \alpha, \beta \geq 0 \\
\quad \quad \text{Let } r(z) \triangleq (g(z), h(z)) \text{ be a continuous two-dimensional curve in } [-1, \alpha] \times [-1, \beta]; \\
\quad \quad \text{Compute } \text{conc-env}(r) \text{ (i.e. positive-orthant concave envelope of } r(t) \text{ as in Definition 3);} \\
\quad \quad \text{Find point } P \triangleq \text{intersection of } \text{conc-env}(r) \text{ and the line } h' - \beta = g' - \alpha \text{ on g-h plane;} \\
\quad \quad \text{Suppose } P = \lambda P_1 + (1 - \lambda) P_2, \text{ where } \lambda \in [0, 1] \text{ and } P_j = r(z^{(j)}), z^{(j)} \in [Z_l, Z_u] \text{ for } j = 1, 2, \text{ and both points are also on the } \text{conc-env}(r); \text{ // see Figure 2} \\
\quad \quad \text{Randomly pick } \hat{z}_i \text{ such that} \\
\quad \quad \quad \begin{cases} 
\hat{z}_i \leftarrow z^{(1)} \text{ with probability } \lambda, \\
\hat{z}_i \leftarrow z^{(2)} \text{ o.w.} 
\end{cases}; \\
\quad \text{Let } X_i \leftarrow \hat{z}_i \text{ and } Y_i \leftarrow \hat{z}_i; \text{ // after this, } X \text{ and } Y \text{ will agree at coordinate } i \\
\end{algorithm}

\textbf{Proof sketch.} Our bi-greedy approach, à la Buchbinder et al. [2012, 2015], revolves around analyzing the evolving values of three points: $X^{(i)}$, $Y^{(i)}$, and $O^{(i)}$. These three points begin at all-zeroes, all-ones, and the optimum solution, respectively, and converge to the algorithm’s final point. In each iteration, we aim to relate the total increase in value of the first two points with the decrease in value of the third point. If we can show that the former quantity is at least twice the latter quantity, then a telescoping sum proves that the algorithm’s final choice of point scores at least half that of optimum.

The utility of our game is specifically engineered to compare the total increase in value of the first two points with the decrease in value of the third point. If we show that the former quantity is at least twice the latter quantity, then a telescoping sum proves that the algorithm’s final choice of point scores at least half that of optimum.

\textbf{Analyzing the Zero-sum Games.} Fix an iteration $i \in [n]$ of Algorithm 1. We then have the following.

\textbf{Proposition 1.} If ALG plays the (randomized) strategy $\hat{z}_i$ as described in Algorithm 1, then we have $E \left[ \mathcal{Y}^{(i)}(\hat{z}_i, x^*_i) \right] \geq 0 \text{ against any strategy } x^*_i \text{ of } \text{ADV.}$

\textbf{Proof of Proposition 1.} We do the proof by case analysis over two cases:

\textbf{Case Z}_1 \geq Z_u \text{ (easy):} See the supplementary materials for this case.

\textbf{Case Z}_1 < Z_u \text{ (hard):} In this case, ALG plays a mixed strategy over two points. To determine the two-point support, it considers the curve $r = \{(g(z), h(z))\}_{z \in [Z_l, Z_u]}$ and finds a point $P$ on \text{conc-env}(r) (i.e., Definition 3) that lies on the line $h' - \beta = g' - \alpha$, where recall that $\alpha = g(Z_u) \geq 0$ and $\beta = g(Z_l) \geq 0$ (as $Z_u$ and $Z_l$ are the maximizers of $\mathcal{F}(z, X^{(i-1)})$ and $\mathcal{F}(z, Y^{(i-1)})$ respectively). Because this point is on the concave envelope it should be a convex combination of two points on the curve $r(z)$. Let $\lambda = 0$ or $\lambda = 1$, where $P_1 = r(z^{(1)})$ and $P_2 = r(z^{(2)})$, and $\lambda \in [0, 1]$. The final strategy of ALG is a mixed strategy over $\{z^{(1)}, z^{(2)}\}$ with probabilities $\lambda, 1 - \lambda$. Fixing any mixed strategy of ALG over two points $P_1 = (g_1, h_1)$ and $P_2 = (g_2, h_2)$ with probabilities
(λ, 1 − λ) (denoted by \( F_P \)), define the \( \text{ADV} \)'s positive region, i.e.:

\[
(g', h') \in [-1, 1] \times [-1, 1] : \quad \mathbb{E}_{(g, h) \sim F_p} \left[ \frac{1}{2} g + \frac{1}{2} h - \max(g' - g, h' - h) \right] \geq 0.
\]

Now, suppose \( \text{ALG} \) plays a mixed strategy with the property that its corresponding \( \text{ADV} \)'s positive region covers the entire curve \( \{g(z), h(z)\}_{z \in [0,1]} \). Then, for any strategy \( x_i^* \) of \( \text{ADV} \) the expected utility of \( \text{ALG} \) is non-negative. In the rest of the proof, we geometrically characterize the \( \text{ADV} \)'s positive region against a mixed strategy of \( \text{ALG} \) over any 2-point support, and then we show for the particular choice of \( P_1, P_2 \) and \( \lambda \) in Algorithm 1 the positive region covers the entire curve \( \{g(z), h(z)\}_{z \in [0,1]} \).

Lemma 2. Suppose \( \text{ALG} \) plays a 2-point mixed strategy over \( P_1 = r(\lambda) = (g_1, h_1) \) and \( P_2 = r(\lambda) = (g_2, h_2) \) with probabilities \( (\lambda, 1 - \lambda) \), and w.l.o.g. \( h_1 - g_1 \geq h_2 - g_2 \). Then \( \text{ADV} \)'s positive region is the pentagon \( (\mathcal{M}_0, \mathcal{M}_1, \mathcal{Q}_1, \mathcal{Q}_2, \mathcal{M}_2) \), where \( \mathcal{M}_0 = (-1, -1) \) and (see Figure 2):

1. \( \mathcal{M}_1 = (-1, \lambda(\frac{3}{2}h_1 + \frac{1}{2}g_1) + (1 - \lambda)(\frac{3}{2}h_2 + \frac{1}{2}g_2)) \).
2. \( \mathcal{M}_2 = (\lambda(\frac{3}{2}g_1 + \frac{1}{2}h_1) + (1 - \lambda)(\frac{3}{2}g_2 + \frac{1}{2}h_2), -1) \).
3. \( \mathcal{Q}_1 \) is the intersection of the lines leaving \( P_1 \) with slope 1 and leaving \( \mathcal{M}_1 \) along the g-axis.
4. \( \mathcal{Q}_2 \) is the intersection of the lines leaving \( P_2 \) with slope 1 and leaving \( \mathcal{M}_2 \) along the h-axis.

By applying Lemma 2, we have the following main technical lemma. The proof is geometric and is pictorially visible in Figure 2. This lemma finishes the proof of Proposition 1.

Lemma 3 (main lemma). If \( \text{ALG} \) plays the two point mixed strategy described in Algorithm 1, then for every \( x_i^* \in [0,1] \) the point \( (g', h') = (g(x_i^*), h(x_i^*)) \) is in the \( \text{ADV} \)'s positive region.

Proof sketch. For simplicity assume \( Z_l = 0 \) and \( Z_u = 1 \). To understand the \( \text{ADV} \)'s positive region that results from playing a two-point mixed strategy by \( \text{ALG} \), we consider the positive region that results from playing a one point pure strategy. When \( \text{ALG} \) chooses a point \( (g, h) \), the positive term of the utility is one-half of its one-norm. The negative term of the utility is the worse between how much the \( \text{ADV} \)'s point is above \( \text{ALG} \)'s point, and how much it is to the right of \( \text{ALG} \)'s point. The resulting positive region is defined by an upper boundary \( g' \leq \frac{3}{2}g + \frac{1}{2}h \) and a right boundary \( h' \leq \frac{1}{2}g + \frac{3}{2}h \).

Next, let's consider what happens when we pick point \( (g_1, h_1) \) with probability \( \lambda \) and point \( (g_2, h_2) \) with probability \( (1 - \lambda) \). We can compute the expected point: let \( (g_3, h_3) = \lambda(g_1, h_1) + (1 - \lambda)(g_2, h_2) \). As suggested by Lemma 2, the positive region for our mixed strategy has three boundary conditions: an upper boundary, a right boundary, and a corner-cutting boundary. The first two boundary conditions correspond to a pure strategy which picks \( (g_3, h_3) \). By design, \( (g_3, h_3) \) is located so that these boundaries cover the entire \([-1, \alpha] \times [-1, \beta] \) rectangle. This leaves us with analyzing the corner-cutting boundary, which is the focus of Figure 2. As it turns out, the intersections of this boundary with the two other boundaries lie on lines of slope 1 extending from \( (g_j, h_j)_{j=1,2} \). If we consider the region between these two lines, the portion under the envelope (where the curve \( r \) may lie) is distinct from the portion outside the corner-cutting boundary. However, if \( r \) were to ever violate the corner-cutting boundary condition without violating the other two boundary conditions, it must do so in this region. Hence the resulting positive region covers the entire curve \( r \), as desired.

2.2 Polynomial-time Implementation under Lipschitz Continuity: Overview

At each iteration, Algorithm 1 interfaces with \( \mathcal{F} \) in two ways: (i) when performing optimization to compute \( Z_l, Z_u \) and (ii) when computing the upper-concave envelope. In both cases, we are concerned with univariate projections of \( \mathcal{F} \), namely \( \mathcal{F}(z, X_{-i}) \) and \( \mathcal{F}(z, Y_{-i}) \). Assuming \( \mathcal{F} \) is coordinate-wise Lipschitz continuous with constant \( C > 0 \), we choose a small \( \epsilon > 0 \) and take periodic samples at \( \epsilon \)-spaced intervals from each one of these functions, for a total of \( O(\frac{1}{\epsilon}) \) samples.

To perform task (i), we simply return the the sample which resulted in the maximum function value. Since the actual maximum is \( \epsilon \)-close to one of the samples, our maximum is at most an additive \( \epsilon C \) lower in value. To perform task (ii), we use these samples to form an approximate \( r(z) \) curve, denoted by \( \hat{r}(z) \). Note that we then proceed exactly as described in Algorithm 1 to pick a (randomized) strategy.
\[ \hat{z}_i \text{ using } \hat{r}(z). \] Note that \( \text{ADV} \) can actually choose a point on the exact curve \( \hat{r}(z) \). However the point she chooses is close to one of our samples and hence is at most an additive \( \epsilon C \) better in value with respect to functions \( g(\cdot) \) and \( h(\cdot) \). Furthermore, we can compute the upper-concave envelope \( \hat{r}(z) \) in time linear in the number of samples using Graham’s algorithm [Graham, 1972]. Roughly speaking, this is because we can go through the samples in order of \( z \)-coordinate, avoiding the sorting cost of running Graham’s on completely unstructured data. Formally, we have the following proposition. See the supplementary materials for detailed implementations (Algorithm 3 and Algorithm 4).

**Proposition 2.** If \( \mathcal{F} \) is coordinate-wise Lipschitz continuous with constant \( C > 0 \), then Algorithm 1 can be implemented with \( O(n^2/\epsilon) \) calls to \( \mathcal{F} \) and returning a (randomized) point \( \hat{z} \) s.t.

\[ 2E[\mathcal{F}(\hat{z})] \geq \mathcal{F}(x^*) - 2C\epsilon, \]  
where \( x^* = \arg\max_{x \in [0,1]^n} \mathcal{F}(x) \) is the optimal solution.

### 3 Strong DR-SM Maximization: Binary-Search Bi-Greedy

Our second result is a fast binary search algorithm, achieving the tight \( 1/2 \)-approximation factor (up to additive error \( \delta \)) in quasi-linear time in \( n \), but only for the special case of strong DR-SM functions (a.k.a. DR-submodular); see Definition 1. This algorithm leverages the coordinate-wise concavity to identify a coordinate-wise monotone equilibrium condition. In each iteration, it hunts for an equilibrium point by using binary search. Satisfying the equilibrium at each iteration then guarantees the desired approximation factor. Formally we propose Algorithm 2. As a technical assumption, we assume \( \mathcal{F} \) is Lipschitz continuous with some constant \( C > 0 \), so that we can relate the precision of our binary search with additive error. We arrive at the theorem, whose proof is in the supplement.

**Theorem 2.** If \( \mathcal{F}(\cdot) \) is non-negative and DR-submodular (a.k.a Strong DR-SM) and is coordinate-wise Lipschitz continuous with constant \( C > 0 \), then Algorithm 2 runs in time \( O(n \log (\frac{n}{\epsilon})) \) and is a deterministic \( 1/2 \)-approximation algorithm up to \( O(\epsilon) \) additive error, i.e. returns \( \hat{z} \in [0,1]^n \) s.t.

\[ 2\mathcal{F}(\hat{z}) \geq \mathcal{F}(x^*) - 2C\epsilon, \]  
where \( x^* = \arg\max_{x \in [0,1]^n} \mathcal{F}(x) \) is the optimal solution.

**Running Time.** If we show that \( f(z) \triangleq \frac{\partial^2 \mathcal{F}}{\partial z_i^2}(z, X_{-i})(1 - z) + \frac{\partial \mathcal{F}}{\partial z_i}(z, Y_{-i})z \) is monotone non-increasing in \( z \), then clearly the binary search terminates in \( O(\log(n/\epsilon)) \) steps (note that the algorithm only does binary search in the case when \( f(0) > 0 \) and \( f(1) < 0 \). To see the monotonicity,

\[ f'(z) = (1 - z) \frac{\partial^2 \mathcal{F}}{\partial z_i^2}(z, X_{-i}) + z \frac{\partial^2 \mathcal{F}}{\partial z_i^2}(z, Y_{-i}) + \left( \frac{\partial \mathcal{F}}{\partial z_i}(z, Y_{-i}) - \frac{\partial \mathcal{F}}{\partial z_i}(z, X_{-i}) \right) \leq 0 \]
where the inequality holds due to strong DR-SM and the fact that all of the Hessian entries (including diagonal) are non-positive. Hence the total running time is $O(n \log(n/\epsilon))$.

4 Experimental Results

We empirically measure the solution quality of three algorithms: Algorithm 1 (GAME), Algorithm 2 (BINARY) and the Bi-Greedy algorithm of Bian et al. [2017b] (BMBK). These are all based on a double-greedy framework, which we implemented to iterate over coordinates in a random order. These algorithms also do not solely rely on oracle access to the function; they invoke one-dimensional optimizers, concave envelopes, and derivatives. We implement the first and the second (Algorithm 3 and Algorithm 4 in the supplement), and numerically compute derivatives by discretization. We consider two application domains, namely Non-concave Quadratic Programming (NQP) [Bian et al., 2017b, Kim and Kojima, 2003, Luo et al., 2010], under both strong-DR and weak-DR, and maximization of softmax extension for MAP inference of determinantal point process [Kulesza et al., 2012, Gillenwater et al., 2012]. Each experiment consists of twenty repeated trials. For each experiment, we use $n = 100$ dimensional functions. Our experiments were implemented in python. See the supplementary materials for the detailed specifics of each experiment. The results of our experiments are in Table 1, and the corresponding box and whisker plots are in Figure 3. The data suggest that for all three experiments the three algorithms obtain very similar objective values. For example, in the weak-DR NQP experiment, all three algorithms have standard deviations around 6 while their means differ by less than 1.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>NQP $\forall_{i,j} : H_{i,j} \leq 0$ (strong-DR)</th>
<th>NQP $\forall_{i} \neq j : H_{i,j} \leq 0$ (weak-DR)</th>
<th>Softmax Ext. (strong-DR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAME</td>
<td>1225.416454 ± 8.201871</td>
<td>1200.860403 ± 6.009484</td>
<td>24.056934 ± 3.794209</td>
</tr>
<tr>
<td>BINARY</td>
<td>1225.392136 ± 8.203827</td>
<td>1200.248876 ± 6.088293</td>
<td>23.945428 ± 3.770932</td>
</tr>
<tr>
<td>BMBK</td>
<td>1225.339063 ± 8.141104</td>
<td>1200.798114 ± 5.975035</td>
<td>24.055435 ± 3.796350</td>
</tr>
</tbody>
</table>

Table 1: Experimental results listing mean and standard deviation over $T = 20$ repeated trials with dimension $n = 100$.

5 Conclusion

We proposed a tight approximation algorithm for continuous submodular maximization, and a quasilinear time tight approximation algorithm for the special case of DR-submodular maximization. Our experiments also verify the applicability of these algorithms in practical domains in machine learning. One interesting avenue for future research is to generalize our techniques to the maximization over any arbitrary separable convex set, which will result in a broader application domain.
References


