The Adaptive Complexity of Maximizing a Gross Substitutes Valuation

Ron Kupfer
The Hebrew University of Jerudalem
ron.kupfer@mail.huji.ac.il

Eric Balkanski
Harvard University
ericbalkanski@g.harvard.edu

Sharon Qian
Harvard University
sharonqian@g.harvard.edu

Yaron Singer
Harvard University
yaron@seas.harvard.edu

Abstract

In this paper, we study the adaptive complexity of maximizing a monotone gross substitutes function under a cardinality constraint. Our main result is an algorithm that achieves a $1-\epsilon$ approximation in $\mathcal{O}(\log n)$ adaptive rounds for any constant $\epsilon > 0$, which is an exponential speedup in parallel running time compared to previously studied algorithms for gross substitutes functions. We show that the algorithmic results are tight in the sense that there is no algorithm that obtains a constant factor approximation in $\tilde{o}(\log n)$ rounds. Both the upper and lower bounds are under the assumption that queries are only on feasible sets (i.e., of size at most k). We also show that under a stronger model, where non-feasible queries are allowed, there is no non-adaptive algorithm that obtains an approximation better than $1/2+\epsilon$. Both lower bounds extend to the class of OXS functions. Additionally, we conduct experiments on synthetic and real data sets to demonstrate the near-optimal performance and efficiency of the algorithm in practice.

1 Introduction

In this paper, we study the problem of maximizing gross substitutes functions in the adaptive complexity model. Gross substitutes are an extremely well-studied class of functions in microeconomics. The concept of gross substitutes was first introduced in the seminal work by Arrow and Debreu as a sufficient condition on the valuation functions of buyers to guarantee the existence of equilibria in markets with indivisible items [1]. It was later shown to also be a necessary condition [20]. Gross substitutes functions are also studied in the contexts of stable matchings in two-sided markets [2, 33], combinatorial auctions [3], and trading networks [23, 25], and have been rediscovered in multiple fields under different names. We refer the reader to [31] for a survey of the different definitions.

In theoretical computer science and optimization, gross substitutes are considered as a subclass of *submodular functions*, as they satisfy the diminishing returns property. For monotone submodular functions, it is well known that a greedy algorithm that iteratively selects the element with the maximal marginal contribution to its current solution obtains a $1-1/e$ approximation for maximization under a cardinality constraint [30] and that this bound is optimal for polynomial-time algorithms [29, 19]. For gross substitutes functions, the greedy algorithm returns an *optimal* solution [13]. Thus, from a purely algorithmic perspective, gross substitutes represent an important subclass of submodular functions: it is the most expressive class of submodular functions that can be optimized *exactly* under cardinality constraints in polynomial time. Not only is gross substitutability a sufficient condition for the optimality of greedy, but it is also a necessary condition [31].[1]

[1] In the context of Walrasian equilibrium, gross substitutes correspond exactly to the class of functions for which the greedy algorithm is optimal for all price vectors [31].

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

A recent line of work began investigating the *adaptive complexity* of submodular optimization [8, 6, 15, 11, 18, 9, 5, 17, 12, 7, 10, 16, 26]. The adaptive complexity model was introduced in [8] as an information theoretic measure for the parallel runtime of an algorithm. Informally, the adaptivity of an algorithm is its number of sequential rounds, when each round can perform polynomially-many function evaluations in parallel. Since the greedy algorithm adds a single element to the current solution at every iteration, it has adaptivity that is linear in the cardinality constraint k, which, in the worst case, is $\Omega(n)$. Until recently, there was no known constant factor approximation algorithm whose adaptivity is sublinear in k for maximizing a submodular, or even a gross substitutes function.

The main result in [8] is an algorithm that obtains a constant factor approximation arbitrarily close to $1/3$ in $\mathcal{O}(\log n)$ rounds, which was an exponential speedup in parallel runtime for maximizing monotone submodular functions under a cardinality constraint. They also showed that there is no $\tilde{o}(\log n)$ adaptive algorithm that achieves a constant approximation. The algorithm in [8] uses a technique called adaptive sampling that was first extended by [6, 15] to obtain an approximation that is arbitrarily close to the optimal $1 - 1/e$, and then by other papers in this genre [18, 5, 17, 26].

Although there has been a great deal of work on submodular maximization in the adaptive complexity model, this work has focused on general submodular functions and little is known about the adaptive complexity of maximizing gross substitutes functions. On one hand, gross substitutes are a superclass of additive and unit-demand functions. For additive and unit demand algorithms, it is trivial to obtain an approximation arbitrarily close to optimal (i.e. $1 - \epsilon$, for any constant ϵ) with only 1 round. On the other hand, gross substitutes are a *subclass* of submodular functions. Thus, all the results in the adaptive complexity model for submodular functions also apply to gross substitutes and there is a $\mathcal{O}(\log n)$-adaptive algorithm that obtains an approximation arbitrarily close to $1 - 1/e$ for maximizing monotone gross substitutes under a cardinality constraint. But to obtain near optimal results, the only algorithm known is the greedy algorithm whose adaptivity is linear in k.

How many rounds are needed to find a (near) optimal solution to the problem of maximizing gross substitutes under a cardinality constraint?

Main results. We first show that the number of rounds needed to find a solution that is arbitrarily close to optimal for maximizing monotone gross substitutes under a cardinality constraint is $\mathcal{O}(\log n)$. In particular, for any $\epsilon > 0$, there exists an $\mathcal{O}(\log(n)/\epsilon^3)$ adaptive algorithm that obtains a $1 - \epsilon$ approximation in expectation. This near-optimal algorithm provides an exponential improvement in parallel runtime compared to previous algorithms for maximizing gross substitutes. We also provide two lower bounds. The first shows that there is no non-adaptive, i.e. 1-adaptive, algorithm that obtains an approximation better than $1/2 + \epsilon$ for maximizing monotone gross substitutes under a cardinality constraint. This hardness result shows a sharp separation between gross substitutes and additive and unit demand functions which can both be optimized arbitrarily well in a single round. The second lower bound is a conditional lower bound. Assuming that the algorithm queries sets of size $\mathcal{O}(k)$, there is no $\tilde{o}(\log n)$ algorithm that obtains a constant approximation for maximizing monotone gross substitutes under a cardinality constraint.

Furthermore, we conduct experiments on synthetic bipartite graphs and Twitter data. We observe that the algorithm has near-optimal performance while running in exponentially fewer parallel rounds. Additionally, the adaptive algorithm outperforms its benchmarks on a range of different valuations. In practice, we observe that the true number of rounds the algorithm requires is much lower than the theoretical bound for the approximation guarantee.

1.1 Technical Overview

The algorithm. To show low adaptivity and near-optimal approximation, we first define two classes of algorithms called impatient greedy and stochastic greedy. Each iteration of a stochastic greedy algorithm selects an element whose marginal contribution is in expectation a $1 - \epsilon$ approximation to the optimal marginal contribution at that iteration. We show that this algorithm obtains a $1 - \epsilon$ approximation for gross substitutes.

Most low adaptivity algorithms for submodular functions use a technique called adaptive sampling [8, 15, 18, 17, 5, 6, 26]. Unfortunately, adaptive sampling does not guarantee that elements added to the solution have near-optimal contribution at each iteration, which fails to give near-optimal guarantees for gross substitutes (see example in Appendix A.2). Instead, the algorithm leverages a recent

adaptive sequencing technique [7] that guarantees near-optimality of the marginal contributions of elements added. However, adaptive sequencing from [7] has $\mathcal{O}(\log(n)\log(k))$ adaptivity.

To improve the adaptivity, we introduce the class of impatient greedy algorithms, which begin by adding elements as long as their marginal contributions are above some fixed threshold, i.e., not necessarily close to optimal. We show that with threshold $\frac{\texttt{OPT}}{\epsilon k}$, impatient greedy algorithms obtain near-optimal approximation guarantees for gross substitutes. The main algorithm employs the adaptive sequencing technique starting with a threshold equal to $\frac{\texttt{OPT}}{\epsilon k}$. With this low threshold, adaptivity is improved from $\mathcal{O}(\log(n)\log(k))$ to $\mathcal{O}(\log n)$ with an arbitrarily small loss in the approximation. In this paper, we only consider maximization under cardinality constraint and leave general matroid constraints for future work. Since the greedy algorithm does not yield the optimal solution for gross substitutes under matroid constraints, many of these techniques cannot be immediately extended.

Lower bounds. Previous lower bound constructions in the adaptive complexity model [8, 9] are submodular, but not gross substitutes, so novel constructions are needed. The functions we construct to be hard to optimize are OXS functions, thus our lower bounds also hold for this subclass of gross substitute functions. The main challenge in the analysis of $\tilde{o}(\log n)$ rounds lower bound is handling the subtle interactions between the different rounds of queries by the algorithm. Our approach is related to the round elimination technique from communication complexity.

2 Preliminaries

We assume value oracle access to a function $f : 2^N \to \mathbb{R}$. An algorithm is *r-adaptive* if it consists of r sequential rounds where the algorithm may perform $\text{poly}(n)$ function evaluations $f(S)$ in parallel at every round. A function f is *submodular* if it exhibits the diminishing returns property, i.e., $f_S(a) \geq f_T(a)$ for all $S \subseteq T \subseteq N$ and $a \in N \setminus T$, where $f_S(T) := f(S \cup T) - f(S)$ is the marginal contribution of T to S. We abuse notation and write $f(a)$, $f_S(a)$ for $f(\{a\})$ and $f_S(\{a\})$ when clear from context. As discussed in the introduction, there exist many equivalent definitions for gross substitutes (GS), see [31] for a detailed survey. We give the definition which we use for the analysis. A function f is *gross substitutes* if it is submodular and for all $S, T \subseteq N$ and $a \in S$, $f(S) + f(T) \leq \max_{b \in T} \{f(S \setminus a) + f(T \cup a), f(S \cup b \setminus a) + f(T \cup a \setminus b)\}$. We show our lower bounds for gross substitutes by constructing families of OXS functions. A function f with $N = \{a_1, \ldots, a_n\}$ is a unit-demand function if there are n positive weights $w_1, \ldots, w_n \in \mathbb{R}_+$ s.t. $f(S) = \max_{a_i \in S} w_i$. A function f is an *assignment function (OXS)* if f is the convolution of r unit-demand functions $u_1, \ldots, u_r$: $f(S) = \bigvee_{i \in [r]} u_i(S) := \max_{\cup_{i \in [r]} S_i = S} \sum_{i \in [r]} u_i(S_i)$ where sets $S_1, \ldots S_r$ are a partition of S. These three classes are related as follows [27]: $OXS \subsetneq GS \subsetneq SM$.

3 $\mathcal{O}(\log n)$ Rounds Suffice for Near Optimal Approximation

In this section we describe an algorithm for maximizing gross substitutes functions which has low adaptivity and returns a solution whose approximation guarantee is arbitrarily close to optimal. To prove these properties we first define two classes of algorithms – *impatient greedy* and *stochastic greedy* algorithms. We show that impatient greedy algorithms yield low adaptivity algorithms for gross substitutes functions and that stochastic greedy algorithms yield approximately optimal algorithms. We then define our main algorithm which is both an impatient greedy algorithm and a stochastic greedy algorithm and can, therefore, instantiate the guarantees for both to show that the algorithm is $\mathcal{O}(\log n)$ adaptive and achieves an approximation guarantee arbitrarily close to optimal.

3.1 Impatient Greedy Analysis

An impatient algorithm first collects items with marginal contribution to the current set above some input threshold t. In the second stage, the algorithm adds the remaining elements using the greedy algorithm. We show that by choosing the correct threshold t, this algorithm performs nearly optimally for gross substitutes functions. This choice of the threshold is the crucial step in showing a good approximation guarantee is achievable in few rounds. All proofs are deferred to Appendix B.

With threshold $t = \frac{\texttt{OPT}}{\epsilon k}$, we show that stochastic greedy performs near optimally for gross substitutes. This follows from the fact that the number of elements chosen in the first loop is bounded by ϵk (Lemma 3). We defer the analogous result for submodular functions to Appendix B.3.

Algorithm 1 IMPATIENT GREEDY

Input $f(\cdot),\ k,\ t$
$S \leftarrow \emptyset,\ X \leftarrow N$
while $X \neq \emptyset$ and $|S| < k$ **do**
    $X \leftarrow \{a : f_S(a) \geq t\}$
    $S \leftarrow S \cup \{a_i\}$ where a_i is chosen u.a.r. from X
while $|S| < k$ **do**
    $X \leftarrow \{a : f_S(a) = \max_x f_S(x)\}$
    $S \leftarrow S \cup \{a_i\}$ where a_i is chosen arbitrarily from X
return S

Theorem 1. *Given a monotone gross substitutes function $f : 2^N \to \mathbb{R}$,* IMPATIENT GREEDY *with threshold $t = \frac{\texttt{OPT}}{\epsilon k}$ returns a set S such that $f(S) \geq (1-\epsilon)\texttt{OPT}$.*

3.2 STOCHASTIC GREEDY Analysis

In this section, we show that the *stochastic greedy* algorithm can guarantee a strong approximation to the optimal solution of gross substitutes functions. At each step i of the algorithm, noise parameters are sampled from distribution $\mathcal{D}_i$ and an approximate maximal element is chosen (Algorithm 2). For submodular functions, stochastic greedy algorithms give approximations arbitrarily close to $1-1/e$ with high probability when k is sufficiently large [22]. Noisy versions of the greedy algorithm on submodular functions are a vast area of research [32, 24, 21, 34, 22]. However, for gross substitutes functions, this has not been studied.

We prove results for gross substitutes functions, which may be of interest even outside the context of adaptive complexity. In particular, we show that for gross substitutes functions, an algorithm that selects an element which is, in expectation, an α approximation to the element with the largest marginal contribution at each iteration provides an α approximation to the optimal solution. This idea is crucial in proving our main result in Theorem 3.

Algorithm 2 STOCHASTIC GREEDY

$S \leftarrow \emptyset$
for $i \in [k]$ **do**
    $(\xi_i, \zeta_i) \sim \mathcal{D}_i$
    $X = \{a : f_S(a) \geq \xi_i \max_x f_S(x) - \zeta_i\}$
    $S \leftarrow S \cup \{a_i\}$ where a_i is chosen u.a.r. from X
return S

We first state the following lemma from [31] which will be useful for bounding the marginal contribution obtained at each step of the algorithm. All proofs are deferred to Appendix C.

Lemma 1 ([31])**.** *Let f be a gross substitutes function, then $\forall S, T \subseteq [n]$ with $|S| = |T|$ and $s \in S \setminus T$, we have $f(S) + f(T) \leq \max_{t \in T \setminus S} \{f(S \cup t \setminus s) + f(T \cup s \setminus t)\}$.*

We can use this to show that at any iteration, there exists an element with high marginal contribution to the current solution (Lemma 5) and obtain the following approximation guarantee.

Theorem 2. *Given a gross substitutes function $f : 2^N \to \mathbb{R}$, let S be the set of size k selected by* STOCHASTIC GREEDY *with $\hat{\xi} = \min_i \mathbb{E}[\xi_i]$ and $\hat{\zeta} = \sum_i \mathbb{E}[\zeta_i]$. Then $\mathbb{E}[f(S)] \geq \hat{\xi}\texttt{OPT} - \hat{\zeta}$.*

It now follows that the stochastic variant of the greedy algorithm gives a good approximation to the maximal value when $\hat{\xi} \approx 1$ and $\hat{\zeta} \approx 0$. In the case where the expected noise is bounded and $\mathbb{E}[\xi_i] = 1-\epsilon$ and $\zeta_i = 0$ for all i, we can get a good approximation to the optimal solution in expectation, i.e. $\mathbb{E}[f(S)] \geq (1-\epsilon)\texttt{OPT}$.

This is indeed the case for the algorithm discussed in the next section. We note that for submodular functions, while the approximation ratio of $1-1/e$ is preserved under noise, it is not always true that the noisy output is close to the output of the greedy algorithm. See example in Appendix A.1.

3.3 The Low Adaptivity Algorithm for Gross Substitutes

We now describe an algorithm for maximizing gross substitutes functions, GROSS SUBSTITUTES ADAPTIVE SEQUENCING (GSAS), which has $\mathcal{O}(\log n)$ rounds and returns a solution whose approximation guarantee is arbitrarily close to optimal. In the analysis, we show how to exploit the guarantees of the two variants presented previously by constructing an algorithm with similar approximation guarantees using a small number of adaptive query rounds. We analyze the performance for both submodular and gross substitutes monotone functions and show that for gross substitutes functions a $1 - \mathcal{O}(\epsilon)$ approximation is obtained[2]. All proofs are deferred to Appendix D.

Adaptive sequencing. In this paper we develop an algorithm that is based on the *adaptive sequencing technique* recently proposed in [7] which was developed to obtain constant factor approximation guarantees under matroid constraints. The overwhelming majority of low-adaptivity algorithms use a different technique called *adaptive sampling* [8, 15, 18, 17, 5, 6, 26], which was introduced in [8]. Adaptive sampling algorithms sample a large number of sets of elements at every iteration to estimate marginal contributions. These estimates, which rely on concentration arguments, are then used to either add a random set R to S or discard elements with low expected contribution to $R \cup S$. Since gross substitutes functions are submodular, adaptive sampling provides a $1 - 1/e$ approximation but fails to give near-optimal guarantees (see Appendix A.2 for an example).

In contrast to adaptive sampling, adaptive sequencing techniques generate at every iteration a *single* random sequence $(a_1, \ldots, a_{|X|})$ of the elements X not yet discarded. Let $A_l = (a_1, \ldots, a_l)$ be a sequence of elements. A prefix $A_{i^\star} = (a_1, \ldots, a_{i^\star})$ of the sequence is then added to the solution S, where $i^\star$ is the largest position i such that a large fraction of the elements in X has high contribution to $S \cup A_{i-1}$. Elements with low contribution to the new solution S are then discarded from X.

Algorithm 3 GROSS SUBSTITUTES ADAPTIVE SEQUENCING (GSAS)

Input $f(\cdot)$, ϵ, Δ, $t^\star$
$S \leftarrow \emptyset$, $t \leftarrow t^\star$
for Δ iterations **do**
  $X \leftarrow N$
  while $X \neq \emptyset$ **do**
    $a_1, ..., a_{k^*} \leftarrow$ random sampling from X of size $k^* = \min(k - |S|, |X|)$
    $X_i \leftarrow \left\{a \in X : f_{S \cup \{a_1, \ldots, a_{i-1}\}}(a) \geq t\right\}$ for all $i \in [k^*]$
    $i^* \leftarrow \min\left\{i : |X_i| \leq (1-\epsilon)|X|\right\}$
    $S \leftarrow S \cup \{a_1, ..., a_{i^*-1}\}$
    $X \leftarrow X_{i^*}$
  $t \leftarrow (1-\epsilon)t$
return S

Variants of Greedy. Adaptive sequencing described in [7] can be viewed as a parallel STOCHASTIC GREEDY algorithm. It starts with a high threshold $t^\star$ and lowers the threshold as the algorithm continues. The initial threshold $t^\star$ is set to $\max f(a)$ so that the algorithm will not discard good elements. In the case where $\texttt{OPT} = \max f(a)$, the number of threshold decrements to guarantee a good approximation is $\mathcal{O}(\log(k))$, which results in the total adaptive complexity of $\mathcal{O}(\log(k)\log(n))$. Using the abstraction of IMPATIENT GREEDY, we can set a lower initial threshold $t^\star = \frac{\texttt{OPT}}{\epsilon k}$ in GSAS to reduce complexity. We show that the number of rounds needed by the outer loop will decrease to $\Delta = \mathcal{O}(1/\epsilon^2)$. From Section 3.1, this threshold adjustment does not greatly effect the performance.

Note that GSAS requires the value of $\texttt{OPT}$ to set the initial threshold. We can bypass this by running several estimations for $\texttt{OPT}$ in parallel, setting $\texttt{OPT} \in \left\{(1+\epsilon)^i \max f(a) |\ i \in \left[\frac{\ln n}{\epsilon}\right]\right\}$, where the running time of each will be truncated by Δ iterations. We will show that a close approximation of $\texttt{OPT}$ is sufficient. From now on, we denote the initial value of t as $t^\star$ where $\frac{\texttt{OPT}}{(1+\epsilon)\epsilon k} \leq t^\star \leq \frac{\texttt{OPT}}{\epsilon k}$.

[2]For submodular functions this algorithm obtains a $1 - 1/e - \mathcal{O}(\epsilon)$ approximation to the optimal solution. We give the analysis in Appendix D.4

We now show our main results that GSAS achieves a $1 - \mathcal{O}(\epsilon)$ approximation in $\mathcal{O}(\log n)$ adaptive rounds. We start by showing logarithmic adaptivity. At a high level, low adaptivity follows from the fact that each inner iteration makes at most $\log(n)/\epsilon$ rounds and the outer loop runs Δ times.

Lemma 2. *Given $\epsilon > 0$ and $\Delta = 1/\epsilon^2$,* GSAS *terminates after $\mathcal{O}(\log(n)/\epsilon^3)$ rounds.*

We now outline a proof sketch for gross substitutes maximization with low adaptivity. We defer the analogous result for submodular functions and its proof to Appendix D.4.

Theorem 3. *For any monotone gross substitutes function f and $\epsilon > 0$,* GSAS *is a $\mathcal{O}(\log(n)/\epsilon^3)$ adaptive algorithm that returns a set S such that $\mathbb{E}[f(S)] \geq (1 - \mathcal{O}(\epsilon))\texttt{OPT}$.*

Proof Sketch. Since the initial threshold of GSAS is lowered, we handle the first iteration separately and similarly to how we handle IMPATIENT GREEDY. We then show that remaining iterations behave similarly to STOCHASTIC GREEDY. Let S_1 be the set at the end of the first iteration. Then, either S_1 is optimal or is small in size w.h.p., i.e. $f(S_1) = \texttt{OPT}$ or $|S_1| < 3\epsilon k$ (Lemma 6).

We now consider all remaining iterations and show that GSAS approximately maximizes a surrogate function $g(A) := f(S_1 \cup A)$ over cardinality constraint $k - |S_1|$. After the first iteration, there are no elements with marginal contribution exceeding $\frac{\texttt{OPT}}{\epsilon k}$ and the algorithm can be reduced to the original version in [7] on g after S_1 has been selected. The approximation guarantee follows from the fact that for each iteration, the threshold t is an approximate upper bound on the maximal marginal contribution of an element a to the intermediate solution S, i.e. $t \geq (1-\epsilon)\max_a g_S(a)$ (Lemma 7). It then follows that GSAS behaves similarly to STOCHASTIC GREEDY where $\mathbb{E}[\xi_i] \geq 1 - 2\epsilon$ and $\zeta_i = 0$ (Lemma 8). We can then use Theorem 2 from STOCHASTIC GREEDY to show that indeed S approximately maximizes g with constraint $k - |S_1|$.

Combining this result with the analysis of the first iteration, we show that S combined with S_1 approximately maximizes f with constraint k. Finally, we handle the possibility of early termination. Since we miss at most k elements, we have a loss of at most $tk = \epsilon\texttt{OPT}$. Thus, we get that GSAS gives a $1 - \mathcal{O}(\epsilon)$ approximation. □

The approximation can also hold with high probability using Markov's Inequality by running polynomially many copies of the algorithm and choosing the maximal one.

4 Lower Bounds

In this section, we present lower bounds on the adaptive complexity of maximizing gross substitutes. We first show that there is no 1-adaptive algorithm that obtains a $1/2 + \epsilon$ approximation, for any constant $\epsilon > 0$ (constructions and proofs deferred to Appendix E). This lower bound shows a sharp separation between gross substitutes and additive and unit demand functions, which can be optimized to be arbitrarily close to 1 in just one round.

Theorem 4. *There is no non-adaptive, i.e. 1-adaptive, algorithm that obtains, with probability $\omega(\frac{1}{n})$, a $1/2 + \epsilon$ approximation for maximizing monotone gross substitutes functions under a cardinality constraint, for any constant $\epsilon > 0$.*

We now show a lower bound for algorithms with multiple adaptive rounds. More precisely, we show that there is no $\tilde{o}(\log n)$ adaptive algorithm that obtains a constant approximation for maximizing OXS functions when the queries are of size $\mathcal{O}(k)$. The main challenge in extending the result from one round is handling the subtle interactions between the different rounds of queries.

We now discuss the assumption that the queries are of size $\mathcal{O}(k)$. We first note that in the context of learning or optimization from past observations and decisions, it is natural that past observations and decisions must also be feasible according to the problem constraint, i.e., of size at most k. In addition, we also note that most of the existing algorithms, e.g. greedy and local search, as well as the algorithm from the previous section, only query feasible sets of size at most k.

Theorem 5. *There is no $(\frac{\log n}{4\log(\log n)} - 1)$-adaptive algorithm that obtains, with probability $\omega(\frac{1}{n})$, a $\frac{1}{\log n}$ approximation for maximizing monotone gross substitutes functions under a cardinality constraint when the queries are sets of size $\mathcal{O}(k)$.*

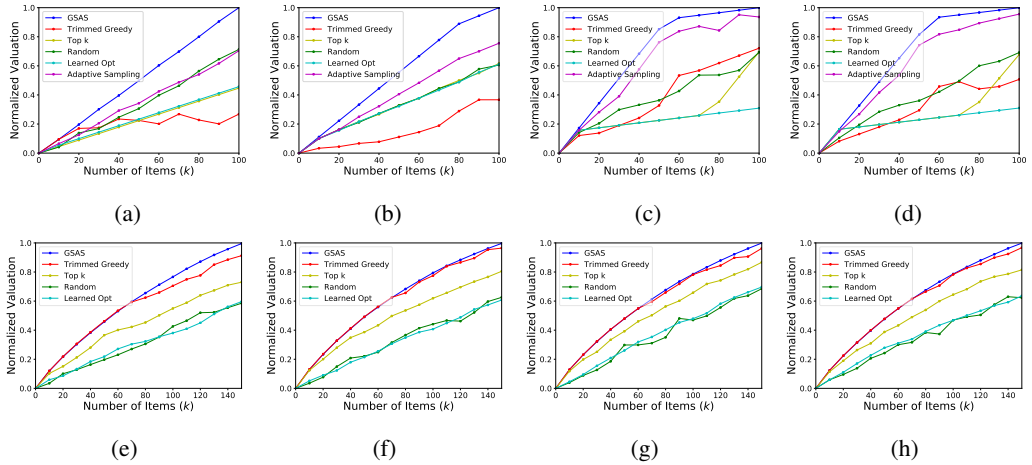


Figure 1: OXS valuation results on synthetic graphs G1, G2, G3 and G4 (Figures 1a, 1b, 1c, 1d) and tweets with #ad, #spon, #giveaway and #win hashtags G5, G6, G7 and G8 (Figure 1e, 1f, 1g, 1h).

5 Experiments

To evaluate the performance of GSAS, we conduct experiments on synthetic and real datasets.

Experimental setup. In our first set of experiments, we construct bipartite graphs with n players, m items (nodes) and valuations (edge weights) to simulate OXS valuations using synthetic and real data. We then compare the performance of GSAS against different benchmarks across varying values of k. We select $k = 100$ elements on synthetic data and $k = 150$ on real data. Our second experiment analyzes the spectrum of unit-demand and additivity. OXS valuations can be represented as the sum of max of item values. In one extreme, valuations are additive so that $v(A) = \sum_{a \in A} v(a)$; in the other extreme, they are unit-demand valuations so that $v(A) = \max_{a \in A} v(a)$. In this experiment, we explore the performance of GSAS by constructing OXS valuations that are strictly unit-demand, additive and in the spectrum and selecting $k = 32$ items for each valuation. For all of the experiments we have used $\epsilon = 0.1$.

Benchmarks. We compare GSAS to the following baselines. **Trimmed Greedy** is the GREEDY algorithm limited by the number of rounds used by GSAS to return a set of size smaller than k. Without this limitation, the algorithm returns the optimal value. **RANDOM** samples, in one round, n many k-tuples and returns the best sample. **TOP-k** selects k elements with largest marginal contribution to empty set in a single round. **Adaptive Sampling** adds sets of elements in each round by iteratively filtering out elements of low marginal contribution and selecting elements of high value [6]. **Learned Opt** first learns the OXS valuation function using sampling and then uses GREEDY to optimize the learned function [4]. We omit `OPT` from our plots since in our experiments `OPT`and GSAS are empirically indistinguishable.

5.1 Datasets

We briefly discuss the generation of synthetic graphs and the constructed Twitter network for the first set of experiments and OXS valuations for the second set. See Appendix F for more details.

Synthetic graphs. We generate the first graph by following the construction detailed in Appendix A.2 (G1) and a second using the construction detailed in Appendix E.3 with $n = 5000$ ground set elements (G2). We construct two additional random bipartite graphs with $n = 275$ ground set items and $m = 200$ players with the probability of an edge fixed at 0.25 (G3) and 0.75 (G4).

Twitter graphs. We filter Twitter data for specific hashtags and extract keywords from each tweet. For each hashtag, we use roughly 500 tweets to construct a bipartite graph with “players" representing advertisements and “items" representing keywords. The valuation of the keyword is determined

by the length of the tweet and the popularity of the keyword. We focus on four different hashtags, each used to construct a different graph: #ad (G5), #spon (G6), #giveaway (G7) and #win (G8).

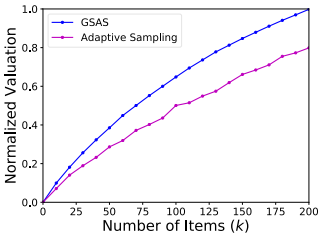
OXS valuations. To analyze the spectrum of unit-demand and additivity, we construct the following OXS valuations. We fix the number of $n = 1024$ items in the ground set. An item of type i, a_j, has value $v(a_j) = i$. We vary the number of item types of items by parameter m, where there are n/m items of each type and each m yields a different OXS valuation. In our construction, all players have the same valuation. In one extreme, for $m = 1$, there is one item type and all 1024 items have value 1, which represents a unit-demand valuation. In the other extreme, $m = 1024$ so that there is exactly one item of each type, which represents an additive function.

5.2 Experimental Results

General performance. Overall, on our synthetic graphs, GSAS (blue) performed near optimally and outperformed the baselines of TRIMMED GREEDY, TOP K, LEARNED OPT and RANDOM (Figures 1a, 1b, 1c, 1d). It was able to obtain high value in much fewer rounds than the traditional GREEDY, which can be seen in the gap of performance between GSAS and TRIMMED GREEDY. We note that LEARNED OPT essentially learns a constant function and cannot distinguish between elements, which causes it to perform similarly to RANDOM.

On the Twitter data, we attempt to select certain advertising tweets that value keywords highly to maximize revenue for an ad placement agency. Sample keywords that were contained in selected ads are listed in Figure 2. We found that for smaller k, GSAS needed as many rounds as GREEDY to terminate so that the performance of both algorithms is near equivalent for k smaller than 80. However, for larger values of k, GSAS terminated in much fewer rounds (Figures 1e, 1f, 1g, 1h).

On Twitter datasets, the oracle call to calculate the OXS valuation is computationally expensive as it includes a maximal weight matching step. Due to computation constraints, we do not use ADAPTIVE SAMPLING, which requires many oracle calls, as a benchmark. In Figure 2, we show the inferior performance of ADAPTIVE SAMPLING compared to GSAS on one such Twitter graph.



#ad	#spon	#giveaway	#win
learn	spiderman	win	case
free	learn	light	light
lives	sign	free	enter
work	free	body	find
win	work	time	free

Figure 2: On the left, GSAS outperforms ADAPTIVE SAMPLING on a Twitter graph. Top keywords for each hashtag are listed on the right.

Fewer number of rounds. We note that in our experimental results, the true number of rounds needed for GSAS to terminate is much lower than the theoretical one of $\log(n)/\epsilon^3$. First, in order to reach value of $t = \epsilon OPT/k$, only $\log(\epsilon^{-1})/\epsilon$ rounds are needed. Second, in the inner loop, the factor of $\log(n)/\epsilon$ is an upper bound and adding more elements into the solution set results in fewer rounds. Additionally, we found that the outer iteration terminated prematurely when the solution set reached k elements. These empirical observations show that GSAS can be much more computationally efficient than GREEDY. Even so, the number of rounds preformed by GSAS was quiet low and presented an improvement in the number of needed rounds. We elaborate on it in Appendix F.

Spectrum of UD-additivity In the two extremes where the OXS valuation is strictly additive or unit-demand (UD), TOP-K performs optimally by selecting the elements with the highest marginal contribution to the empty set in one round. In the general case where the objective valuation lies in the spectrum of UD-additivity, we found that GSAS outperforms its baselines in all regimes. In Figure 3, we normalize all values to the optimal solution as computed by GREEDY. Algorithms that perform better have values close to 1 in the figure.

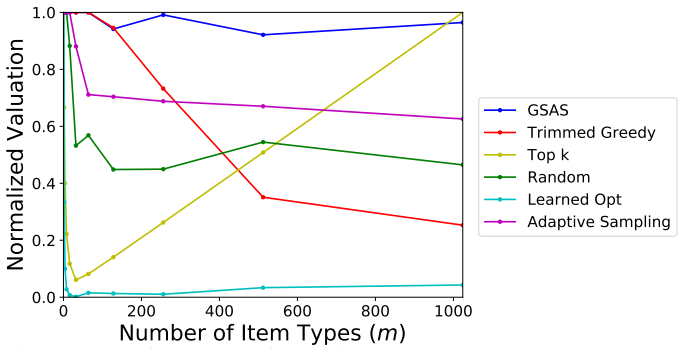


Figure 3: Performance of algorithms on UD-additive valuations.

Broader Impact

This work focuses on the adaptivity of maximizing gross substitutes functions. While previous work has been done on maximizing submodular functions, a superclass of gross substitutes, little is known about the adaptivity complexity to achieve optimal results for this particular class of functions. Our results show an exponentially faster algorithm with near-optimal approximation guarantees for optimization of gross substitute valuations, which have numerous applications in microeconomics and market design [2, 33, 3, 23, 25] and appear in multiple fields such as discrete mathematics [28] and number theory [14].

The algorithm presented in this work is particularly relevant to applications on large datasets where sequential algorithms such as GREEDY become impractical and computationally infeasible. By using a low-adaptivity algorithm such as GSAS, we are able to take advantage of parallelization and dramatically speed up computation on large datasets. In Section 5, we show an application of this algorithm on large constructed Twitter networks to efficiently match keywords in advertisements to bidders or advertisers. In our experimental results, we show both the effective performance and computational efficiency of using GSAS on different networks. This shows that the limited adaptivity of GSAS can be effectively leveraged to analyze trends on other large-scale social networks and applications.

Acknowledgments
Ron Kupfer - This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 740282).
Yaron Singer - This research was supported by BSF grant 2014389, NSF grant CAREER CCF-1452961, NSF USICCS proposal 1540428, Google research award, and a Facebook research award.

References

[1] Kenneth J Arrow and Gerard Debreu. Existence of an equilibrium for a competitive economy. *Econometrica: Journal of the Econometric Society*, pages 265–290, 1954.

[2] John William H as matroid rank functions, gross substitutes. atfield, and Paul R. Milgrom. Matching with Contracts. *American Economic Review*, 95(4):913–935, September 2005.

[3] Lawrence M Ausubel and Paul R Milgrom. Ascending auctions with package bidding. *Advances in Theoretical Economics*, 1(1), 2002.

[4] Maria Florina Balcan, Florin Constantin, Satoru Iwata, and Lei Wang. Learning valuation functions. *arXiv preprint arXiv:1108.5669*, 2011.

[5] Eric Balkanski, Adam Breuer, and Yaron Singer. Non-monotone submodular maximization in exponentially fewer iterations. *NeurIPS*, 2018.

[6] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. An exponential speedup in parallel running time for submodular maximization without loss in approximation. *SODA*, 2019.

[7] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. An optimal approximation for submodular maximization under a matroid constraint in the adaptive complexity model. *STOC*, 2019.

[8] Eric Balkanski and Yaron Singer. The adaptive complexity of maximizing a submodular function. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1138–1151. ACM, 2018.

[9] Eric Balkanski and Yaron Singer. Approximation guarantees for adaptive sampling. In *International Conference on Machine Learning*, pages 393–402, 2018.

[10] Chandra Chekuri and Kent Quanrud. Parallelizing greedy for submodular set function maximization in matroids and beyond. *STOC*, 2019.

[11] Chandra Chekuri and Kent Quanrud. Submodular function maximization in parallel via the multilinear relaxation. *SODA*, 2019.

[12] Lin Chen, Moran Feldman, and Amin Karbasi. Unconstrained submodular maximization with constant adaptive complexity. *STOC*, 2019.

[13] Andreas WM Dress and Werner Terhalle. Well-layered maps?a class of greedily optimizable set functions. *Applied Mathematics Letters*, 8(5):77–80, 1995.

[14] Andreas W.M. Dress and Walter Wenzel. Valuated matroids: a new look at the greedy algorithm. *Applied Mathematics Letters*, 3(2):33 – 35, 1990.

[15] Alina Ene and Huy L Nguyen. Submodular maximization with nearly-optimal approximation and adaptivity in nearly-linear time. *SODA*, 2019.

[16] Alina Ene, Huy L Nguyen, and Adrian Vladu. Submodular maximization with packing constraints in parallel. *STOC*, 2019.

[17] Matthew Fahrbach, Vahab Mirrokni, and Morteza Zadimoghaddam. Non-monotone submodular maximization with nearly optimal adaptivity complexity. *SODA*, 2019.

[18] Matthew Fahrbach, Vahab Mirrokni, and Morteza Zadimoghaddam. Submodular maximization with optimal approximation, adaptivity and query complexity. *SODA*, 2019.

[19] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[20] Faruk Gul and Ennio Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economic Theory*, 87(1):95–124, July 1999.

[21] Avinatan Hassidim and Yaron Singer. Submodular optimization under noise. *arXiv preprint arXiv:1601.03095*, 2016.

[22] Avinatan Hassidim and Yaron Singer. Robust guarantees of stochastic greedy algorithms. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1424–1432. JMLR. org, 2017.

[23] John William Hatfield, Scott Duke Kominers, Alexandru Nichifor, Michael Ostrovsky, and Alexander Westkamp. Stability and competitive equilibrium in trading networks. *Journal of Political Economy*, 121(5):966–1005, 2013.

[24] Thibaut Horel and Yaron Singer. Maximization of approximately submodular functions. In *Advances in Neural Information Processing Systems*, pages 3045–3053, 2016.

[25] Yoshiko T Ikebe, Yosuke Sekiguchi, Akiyoshi Shioura, and Akihisa Tamura. Stability and competitive equilibria in multi-unit trading networks with discrete concave utility functions. *Japan Journal of Industrial and Applied Mathematics*, 32(2):373–410, 2015.

[26] Ehsan Kazemi, Marko Mitrovic, Morteza Zadimoghaddam, Silvio Lattanzi, and Amin Karbasi. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. *ICML*, 2019.

[27] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 18–28. ACM, 2001.

[28] Kazuo Murota and Akiyoshi Shioura. M-convex function on generalized polymatroid. *Mathematics of Operations Research*, 24(1):pp. 95–105, 1999.

[29] George L Nemhauser and Laurence A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978.

[30] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

[31] Renato Paes Leme. Gross substitutability: An algorithmic survey. *Games and Economic Behavior*, 106:294–316, 2017.

[32] Chao Qian, Jing-Cheng Shi, Yang Yu, Ke Tang, and Zhi-Hua Zhou. Subset selection under noise. In *Advances in Neural Information Processing Systems*, pages 3560–3570, 2017.

[33] Alvin E. Roth. Stability and polarization of interests in job matching. *Econometrica*, 52(1):47–57, 1984.

[34] Adish Singla, Sebastian Tschiatschek, and Andreas Krause. Noisy submodular maximization via adaptive sampling with applications to crowdsourced image collection summarization. 2016.

A Constructions

A.1 The Greedy Algorithm is Not Robust to Noise for Submodular Functions

For general submodular functions, it is not always true that the greedy algorithm is robust to noise. Consider the following example from [34]:

$$\begin{gathered} f(\emptyset) = 0 \\ f(\{a\}) = 1,\ f(\{b\}) = 1,\ f(\{c\}) = 1 - \epsilon \\ f(\{a,b\}) = 2,\ f(\{a,c\}) = 1.5 - \tfrac{\epsilon}{2},\ f(\{b,c\}) = 1.5 - \tfrac{\epsilon}{2} \\ f(\{a,b,c\}) = 2 \end{gathered}$$

For $k = 2$, the gap ratio between the noisy and clean versions is $\frac{3}{4}$. In general, it can be as small as $1 - \frac{1}{e}$.

A.2 ADAPTIVE SAMPLING Fails to Maximize Gross Substitutes Functions

In [6], Balkanski et. al. presented an adaptive algorithm for submodular maximization. Similar to the greedy algorithm, this algorithm guarantees a $1 - \frac{1}{e}$ approximation to the optimal value. However, unlike the greedy algorithm, this algorithm does not guarantee an optimal solution for gross substitutes functions.

In this section, we show an example of an OXS function which is difficult for ADAPTIVE SAMPLING when k, the number of elements we wish to select, is small compared to the size of the ground set n.

The construction. We consider the family of partitions $\mathcal{P}$ where the n elements are partitioned into four disjoint sets: A,B,C and D. Formally:

$$\mathcal{P} := \left\{ P = (A,B,C,D) : |A| = \frac{k}{3},\ |B| = |C| = |D| = (n - \frac{k}{3})/3 \right\}.$$

We construct OXS functions f_P which depend on a partition $P \in \mathcal{P}$, so the hard family of OXS functions is

$$\mathcal{F}_1 = \{f_P : P \in \mathcal{P}\}.$$

Given a partition $P = (A,B,C,D)$, we define $\frac{4k}{3}$ unit-demand functions. Those are a_i, b_i, c_i and d_i for $i \in [\frac{k}{3}]$ where

$$a_i(S) = 12 \cdot \mathbb{1}_{|S \cap A| > 0} \quad b_i(S) = 9 \cdot \mathbb{1}_{|S \cap B| > 0} \quad c_i(S) = 6 \cdot \mathbb{1}_{|S \cap C| > 0} \quad d_i(S) = 4 \cdot \mathbb{1}_{|S \cap D| > 0}.$$

The function f_P is then defined to be the OXS function over these unit-demand functions.

Given cardinality constraint k, $\texttt{OPT} = 9k$ simply by taking $k/3$ elements from A, B and C.

For the ADAPTIVE SAMPLING algorithm, assume the number of rounds $r = k$ (which gives the algorithm the highest value). At each round, the algorithm looks for a random set T of size $k/r = 1$ such that

$$f_S(T) = (\texttt{OPT} - f(S))/k$$

where S is the current solution set. In the first round, $S = \emptyset$ and randomly sampled sets should have a marginal contribution of at least $\frac{\texttt{OPT}}{k} = 9$. Since there are very few elements from set A compared to the elements in set B which also have acceptable value, almost none of A will be selected at this step. After $\frac{k}{3}$ elements from set B are added to the solution set, the threshold is now $\frac{\texttt{OPT} - f(S)}{k} = (9k - 9 \cdot \frac{k}{3})/k$. For the same reason as before, the algorithm will now add elements from set C to the solution set. After adding another $\frac{k}{3}$ elements, the threshold is now $\frac{\texttt{OPT} - f(S)}{k} = (9k - 9 \cdot \frac{k}{3} - 6 \cdot \frac{k}{3})/k = 4$. In the last round, $\frac{k}{3}$ elements from set D are have good marginal contribution and with high probability the total value obtained at the end of ADAPTIVE SAMPLING is $\frac{19k}{3}$.

B Missing Proofs for IMPATIENT GREEDY (Section 3.1)

B.1 The First Stage of IMPATIENT GREEDY Selects Few Elements

Lemma 3. *Let $f : 2^N \to \mathbb{R}$ be a submodular function, $t = \frac{\texttt{OPT}}{\epsilon k}$ and S be the set maintained at the end of the first while loop of* IMPATIENT GREEDY. *Then, $|S| \leq \epsilon k$.*

Proof. Denote by $(a_1, ...a_{|S|})$ the elements of S in some order. Since $t = \frac{\texttt{OPT}}{\epsilon k}$, there are at most ϵk elements such that $f_{\{a_1,\ldots,a_{i-1}\}}(a_i) \geq t$ otherwise we have exceeded $\texttt{OPT}$ by selecting this set. □

B.2 Proof of Theorem 1

Proof. Let set S_1 be the set of size $m \leq \epsilon k$ selected in the first while loop and $S_2 = S \setminus S_1$. Let $g(A) = f_{S_1}(A)$ and O be the optimal solution for maximizing f with cardinality constraint of $k - m$. Since g is a gross substitutes function, S_2 is known to be optimal for maximization. Thus,

$$f(S) = f(S_1) + g(S_2) \geq f(S_1) + g(O) = f(S_1 \cup O).$$

By monotonicity, $f(S_1 \cup O) \geq f(O)$ and since $f(O)$ is a $\frac{k-m}{k}$ approximation to $\texttt{OPT}$ we get:

$$f(S) \geq \frac{k-m}{k}\texttt{OPT} \geq \frac{k-\epsilon k}{k}\texttt{OPT} = (1-\epsilon)\texttt{OPT}.$$ □

B.3 IMPATIENT GREEDY for Submodular Functions

Theorem 6. *Given a monotone submodular $f : 2^N \to \mathbb{R}$,* IMPATIENT GREEDY *with threshold $t = \frac{\texttt{OPT}}{\epsilon k}$ returns a set S such that $f(S) \geq (1 - 1/e - \epsilon)\texttt{OPT}$.*

Proof. As before, let set S_1 be the set of size $m \leq \epsilon k$ selected in the first while loop and $S_2 = S \setminus S_1$. Let $g(A) = f_{S_1}(A)$ and O be the optimal solution of size $k - m$ for f. Since g is a monotone submodular function, S_2 is known to guarantee a $1 - \frac{1}{e}$ approximation for its maximization problem. Thus,

$$f(S) = f(S_1 \cup S_2) = f(S_1) + g(S_2) \geq f(S_1) + (1 - e^{-1})g(O) = (1 - e^{-1})(f(S_1) + f_{S_1}(O)).$$

By the monotonicity assumption, $f_O(S_1) \geq 0$ and since $f(O)$ is a $\frac{k-m}{k}$ approximation to $\texttt{OPT}$ we get that

$$f(S_1) + f_{S_1}(O) = f(O) + f_O(S_1) \geq f(O) \geq (1 - \epsilon)\texttt{OPT}.$$

Combining the above results, $f(S) \geq (1 - e^{-1} - \epsilon)\texttt{OPT}$. □

C Missing Proofs for STOCHASTIC GREEDY (Section 3.2)

C.1 Proof of Lemmas for Theorem 2

We first prove the following lemma to compare the solution S to the optimal solution.

Lemma 4. *Given a gross substitutes function $f : 2^N \to \mathbb{R}$ and two sets S, T s.t. $|S| < |T|$, then $f(S) + f(T) \leq \max_{t \in T \setminus S} \{f(S \cup t) + f(T \setminus t)\}$.*

Proof. We define another function f' on $n + |T| - |S|$ items by adding a set D of size $|T| - |S|$ of dummy items: $f'(A \cup R) = f(A)$ for all $R \subseteq D$, where adding R does not affect the function value. It is straightforward to verify that f' is a gross substitutes function. Let $S' = S \cup D$ so $|S'| = |T|$. By Lemma 1 over f', S', T and $s \in D \subseteq S'$, we get

$$f(S)+f(T) = f'(S')+f'(T) \leq \max_{t \in T \setminus S'} \{f'(S' \cup t \setminus s) + f'(T \cup s \setminus t)\} = \max_{t \in T \setminus S} \{f(S \cup t) + f(T \setminus t)\}.$$

□

Lemma 5. *Let $O_i = \{o_1, \ldots, o_i\}$ be the current solution of the greedy algorithm at iteration i (recall that greedy is optimal for gross substitutes) and $S_i = \{a_1, \ldots, a_i\}$ be any set of size i. Then, for all $i < k$, there exists $o \in O_{i+1}$ s.t. $f_{O_i}(o_{i+1}) \leq f_{S_i}(o)$.*

Proof. By Lemma 4 there exists $o \in O_{i+1}$ such that $f(S_i) + f(O_{i+1}) \leq f(S_i \cup o) + f(O_{i+1} \setminus o)$. Rearranging yields the result:

$$f_{O_i}(o_{i+1}) = f(O_{i+1}) - f(O_i) \leq f(O_{i+1}) - f(O_{i+1} \setminus o) \leq f(S_i \cup o) - f(S_i) = f_{S_i}(o)$$

where the first inequality is due to the optimality of O_i and the second from the above inequality. □

C.2 Proof of Theorem 2

Proof. Observe that

$$\mathbb{E}[f(S)] = \mathbb{E}[\sum_{i \leq k} f_{S_{i-1}}(s_i)] \geq \mathbb{E}[\sum_{i \leq k} \xi_i f_{O_{i-1}}(o_i) - \zeta_i] = \sum_{i \leq k} (\mathbb{E}[\xi_i] f_{O_{i-1}}(o_i) - \mathbb{E}[\zeta_i])$$

$$\geq \sum_{i \leq k} \hat{\xi} f_{O_{i-1}}(o_i) - \hat{\zeta} = \hat{\xi}\texttt{OPT} - \hat{\zeta}$$

where the first inequality follows from Lemma 5 and the second from the definitions of $\hat{\xi}$ and $\hat{\zeta}$. □

D Missing Proofs for GSAS (Section 3.3)

D.1 Proof for Lemma 2

Proof. The outer loop runs Δ times. Each inner iteration makes at most $\log(n)/\epsilon$ rounds since at each round, $|X|$ decreases by at least a factor of $1 - \epsilon$. Finding i^* can be done in a single round. □

D.2 Lemmas for the Proof of Theorem 3

Lemma 6. *Given a submodular function $f : 2^N \to \mathbb{R}$, let S be the set at the end of the first while loop of* GSAS. *Then, either $f(S) = \texttt{OPT}$ or $|S| < 3\epsilon k$ with probability $1 - o(e^{\epsilon k/8})$.*

Proof. Denote by $(a_1, ...a_{|S|})$ the elements of S in some order. Since $t^\star$ is a downward discretization of $\frac{\texttt{OPT}}{\epsilon k}$, we have that $t^\star \geq \frac{\texttt{OPT}}{(1+\epsilon)\epsilon k}$ and there are at most $(1+\epsilon)\epsilon k$ elements such that $f_{\{a_1,...,a_{i-1}\}}(a_i) \geq t^\star$ (otherwise we have exceeded OPT by selecting this set). However, we still need to show that the randomization of the algorithm does not choose too many items with marginals contributions below threshold t. Let $a_1, ..., a_l$ be the elements chosen in the process. In order to estimate the number of selected elements such that $f_{\{a_1,...,a_{i-1}\}}(a_i) < t^\star$, we first note that at most k elements are chosen in total. By definition of i^*, for all $i < i^*$, each item a_i has a probability of at least $1 - \epsilon$ to maintain $f_{\{a_1,...,a_{i-1}\}}(a_i) \geq t^\star$. Since each element in the sequence is drawn independently of previous selections, using the Chernoff bound, with probability at least $1 - e^{-\frac{\epsilon k}{8}}$, no more than $1.5\epsilon k$ such elements violate this condition. Thus, at the end of the first iteration, at most $((1+\epsilon) + 1.5)\epsilon$ elements were chosen and $|S| < 3\epsilon k$. □

In the following executions of the outer loop, there are no elements with marginal contribution exceeding $\frac{\texttt{OPT}}{\epsilon k}$ and the algorithm can be reduced to the original version in [7] on f where S is chosen in the first iteration. Below, we show analogous properties but with slight adjustments.

Lemma 7. *Let $f : 2^N \to \mathbb{R}$ be a submodular function. At the end of any inner iteration, $t \geq (1 - \epsilon) \max_a f_S(a)$.*

Proof. At the end of the first iteration, $X = \emptyset$. So, for all $a \in N$, a was discarded from X at some previous inner iteration with current solution S' such that $f_{S' \cup \{a_1,...,a_{i^*}\}}(a) < t$ and $S' \cup \{a_1, ..., a_{i^*}\} \subseteq S$. Thus, by submodularity, $f_S(a) < t$. That is, $t \geq (1 - \epsilon) \max_a f_S(a)$ and the property holds at this point.

We now show that this property is maintained through the algorithm when either S or t is updated: First, since we only add elements to S, the property is unaffected by submodularity. Second, assume we update $t' = (1 - \epsilon)t$. By the same claim as in the first iteration, $t \geq (1 - \epsilon) \max_a f_S(a)$. Since $t' = (1 - \epsilon)t$, we get the desired result. □

Lastly, we show that after the first iteration, each element added to the solution by the algorithm is a stochastic greedy step.

Lemma 8. *Let $a_1, ..., a_{k^*}$ be a random sequence during any iteration of the outer-loop except the first. Then, for all $i < i^*$,*

$$\mathbb{E}_{a_i}[f_{S\cup\{a_1,...,a_{i-1}\}}(a_i)] \geq (1-\epsilon)^2 \max_a f_{S\cup\{a_1,...,a_{i-1}\}}(a)$$

Proof. Observe that

$$\mathbb{E}_{a_i}[f_{S\cup\{a_1,...,a_{i-1}\}}(a_i)] \geq \Pr\left(f_{S\cup\{a_1,...,a_{i-1}\}}(a_i) \geq t\right)\cdot t = \frac{|X_{i-1}|}{|X|}\cdot t \geq (1-\epsilon)^2 \max_a f_{S\cup\{a_1,...,a_{i-1}\}}(a)$$

where the first inequality is by definition of expectation, and the second since $i \leq i^\star$ and by Lemma 7. □

D.3 Proof of Theorem 3

Proof. By Lemma 8, GSAS behaves similarly to the STOCHASTIC GREEDY with noise distribution $\mathcal{D}_i$ where $\zeta_i = 0$ and $\mathbb{E}[\xi_i] \geq (1-\epsilon)^2$. The key differences of GSAS are (1) the initial threshold may be too low and (2) the termination of the algorithm before k elements are selected.

We start by handling the first issue. Let S_1 be the set of size m selected by GSAS in the first round, and $g(A) = f(S_1 \cup A)$ be the conditioned function, O be an optimal solution of the maximization of g under cardinality constraint $k-m$ and G be the solution returned by STOCHASTIC GREEDY for g (under cardinality constraint $k-m$) with $\mathbb{E}[\xi_i] \geq 1-2\epsilon$ and $\zeta_i = 0$ for all i.

By Lemma 8 we have that $\mathbb{E}[g(S)] \geq (1-2\epsilon)g(G)$ and by Theorem 2, we have $\mathbb{E}[g(G)] \geq (1-\epsilon)g(O)$. Together, we get that

$$\mathbb{E}[g(S)] \geq (1-3\epsilon)g(O).$$

Since the greedy algorithm is optimal for gross substitutes functions and g is gross substitutes, we have that $g(O)$ is no less than value of any set of that size. In particular, this is true for the set returned by the greedy algorithm on f with $k-m$ cardinality constraint. That is, $g(O) \geq \texttt{OPT}_{k-m}$ where $\texttt{OPT}_{k-m}$ is the optimal value of f with cardinality constraint $k-m$. Combining the above properties yield

$$\mathbb{E}[g(S)] \geq (1-3\epsilon)g(O) \geq (1-3\epsilon)\texttt{OPT}_{k-m} \geq (1-3\epsilon)\frac{k-m}{k}\texttt{OPT}.$$

By Lemma 6, we have that $m = 3\epsilon k$ and we get that S is a $1-6\epsilon$ approximation.

Next we handle the possibility that GSAS terminates with fewer than k items. Since GSAS terminates after Δ iterations with $t < \frac{\epsilon\texttt{OPT}}{k}$, we may not obtain k elements. However, we miss at most k elements where each element contributes at most t to the solution set. Hence, there is a loss of at most $tk = \epsilon\texttt{OPT}$ of the total value. Thus, we get that GSAS gives a $1-7\epsilon$ approximation. □

D.4 GSAS Obtains a $1-1/e-\mathcal{O}(\epsilon)$ Approximation for Submodular Functions

Theorem 7. *Given a monotone submodular function $f : 2^N \to \mathbb{R}$ and for any $\epsilon > 0$,* GSAS *returns a set S such that $\mathbb{E}[f(S)] \geq (1-1/e-\mathcal{O}(\epsilon))\texttt{OPT}$ approximation ratio using $\mathcal{O}(\log(n)/\epsilon^3)$ rounds.*

Proof. Let set S_1 be the set of size m selected by GSAS in the first round. Let $g(A) = f(S_1 \cup A)$ be a submodular function, A be the solution of size $k-m$ for g returned by the greedy algorithm and G be the solution of size $k-m$ for g returned by STOCHASTIC GREEDY with $\mathbb{E}[\xi_i] \geq 1-2\epsilon$ and $\zeta_i = 0$ for all i. Let S be the result of GSAS.

By Theorem 8 we know that $g(S) \geq (1-2\epsilon)g(G)$. By Theorem 2, we have $\mathbb{E}[g(G)] \geq (1-\epsilon)g(A)$. Combining these together, we get $\mathbb{E}[g(S)] \geq (1-\mathcal{O}(\epsilon))g(A)$.

By [22], we know that the greedy algorithm returns a $1-e^{-(1-\epsilon)}$ approximation to the optimal value of g. By submodularity, we know that the returned value is at least as good as the greedy algorithm

on f with $k-m$ cardinality constraint by monotonicity of f. That is $g(A) \geq (1-e^{-(1-\epsilon)})\texttt{OPT}_{k-m}$ where $\texttt{OPT}_{k-m}$ is the optimal value of f with cardinality constraint $k-m$. Together,

$$\mathbb{E}[g(S)] \geq (1-3\epsilon)g(A) \geq (1-3\epsilon)(1-e^{-(1-\epsilon)})\texttt{OPT}_{k-m} \geq (1-3\epsilon)(1-e^{-(1-\epsilon)})\frac{k-m}{k}\texttt{OPT}$$

Setting $m = 3\epsilon k$ as Lemma 6 suggests, we get a $1-1/e-\mathcal{O}(\epsilon)$ approximation.

Finally, since GSAS terminates after Δ iterations with $t = \frac{\texttt{OPT}}{\epsilon k}$, we may not reach a set of size k. However, we miss at most k elements which contribute value at most tk. Thus, we get a $1-1/e-\mathcal{O}(\epsilon)$ approximation. □

E Missing Proofs from Section 4

E.1 Warm-up: Hardness of Optimization in One Round

Before showing the two main lower bounds, we show that there is no 1-adaptive algorithm that obtains a constant approximation for maximizing OXS functions when the queries are of size $\mathcal{O}(k)$. At a high level, we construct a family of functions $\mathcal{F}$ where each function $f_P \in \mathcal{F}$ is defined by a partition P of the ground set of elements N. The central argument for the analysis is that an algorithm cannot learn the partition P from one round of poly-many queries $f_P(S)$ and that the algorithm needs to know P to find a solution S with high value $f_P(S)$.

The construction. We consider the family of partitions $\mathcal{P}$ where the n elements are partitioned into three disjoint sets: the set G of good elements of size $k = |G| = n^{\frac{1}{3}}$, the set B of bad elements of size $n^{\frac{2}{3}}$, and the set M of masking elements of size $n - |G| - |B|$. Formally:

$$\mathcal{P} := \left\{ P = (G, B, M) : |G| = n^{\frac{1}{3}}, |B| = n^{\frac{2}{3}}, |M| = n - |G| - |B| \right\}.$$

We construct OXS functions f_P which depend on a partition $P \in \mathcal{P}$, so the hard family of OXS functions is $\mathcal{F}_1 = \{f_P : P \in \mathcal{P}\}$. Given a partition $P = (G, B, M)$, we define $n^{\frac{1}{3}} + \frac{n^{\epsilon}}{2}$ unit-demand functions. Those are g_i for $i \in [n^{\frac{1}{3}}]$ and b_i for $i \in [\frac{n^{\epsilon}}{2}]$ where

$$g_i(S) = \mathbb{1}_{|S \cap G| > 0} \quad \text{and} \quad b_i(S) = \mathbb{1}_{|S \cap B| > 0}.$$

The function f_P is then defined to be the OXS function over unit-demand functions g_i, b_i. Note that $f_P(G) = n^{1/3}$, $f_P(B) = n^{\epsilon}/2$, and $f_P(M) = 0$. We obtain the following lower bound for the family of OXS functions $\mathcal{F}_1$.

Theorem 8. *There is no 1-adaptive algorithm that obtains, with probability $\omega(\frac{1}{n})$, an $n^{-\frac{1}{3}+\epsilon}$ approximation for maximizing OXS functions under a cardinality constraint when the queries are sets of size $\mathcal{O}(k)$, for any constant $\epsilon > 0$.*

Proof. First, we show that for any constant $c > 0$, good and bad elements are separable with exponentially small probability when using sample sizes smaller than ck. Since we take a sample of size $t \leq ck$, the probability for seeing more than d elements from either G or B is small:

$$\Pr(|(G \cup B) \cap S| > d) < \binom{t}{d}\left(\frac{|G \cup B|}{n}\right)^d$$
$$< \binom{ck}{d}\left(\frac{2n^{\frac{2}{3}}}{n}\right)^d \leq \left(\frac{2ecn^{\frac{1}{3}}n^{\frac{2}{3}}}{nd}\right)^d = \left(\frac{2ec}{d}\right)^d$$

where the last inequality follows from $\binom{a}{b} \leq \left(\frac{ae}{b}\right)^b$ where e is Euler's number. By taking $d = \frac{n^{\epsilon}}{2}$, we get an exponentially small probability of $|(G \cup B) \cap S| > d$. Thus, the marginal value of good or bad elements equals 1 based on the constructed OXS function.

At the end of the sampling stage, we can assume that g_i and b_i are indistinguishable with high probability. Under this assumption, we can bound the probability for selecting more than d objects from G Since the algorithm chooses up to k elements from $G \cup B$ at random:

$$P(|G \cap S| > d) < \binom{k}{d}\left(\frac{|G|}{|G \cup B|}\right)^d < \binom{n^{\frac{1}{3}}}{d}\left(\frac{n^{\frac{1}{3}}}{n^{\frac{2}{3}}}\right)^d < d^{-d}$$

By taking $d = \frac{n^\epsilon}{2}$ for large enough n, we get an exponentially small probability for selecting more than d elements from G. So any algorithm selects w.h.p. at most $\frac{n^\epsilon}{2}$ elements from G and the rest are from B and M which contribute at most $\frac{n^\epsilon}{2}$. Thus, we get that w.h.p. $f(S) < n^\epsilon$. Since $\texttt{OPT} = n^{\frac{1}{3}}$ the claim holds. □

E.2 Analysis for Hardness of Non-Adaptive Algorithms

The construction. We consider the family of partitions $\mathcal{P}$ that consists of all partitions P where the n elements are partitioned into three disjoint sets as follows: set G of good elements of size $\log^3 n$, set B of bad elements of size $\log^6 n$, and set M of masking elements of size $n - |G| - |B|$.

We construct OXS functions f_P which depend on a partition $P \in \mathcal{P}$, so the hard family of OXS functions is $\mathcal{F} = \{f_P : P \in \mathcal{P}\}$. Given a partition P, we define unit-demand functions function u_i for $i \in [|G| + |B|]$, v_i for $i \in [\log n]$, and w_i for $i \in [\log^3 n]$, where

$$u_i(S) = \mathbb{1}_{|S\cap(G\cup B)|>0}, \qquad v_i(S) = 2_{|S\cap(G\cup B)|>0}, \qquad w_i(S) = \max\left(2_{|S\cap G|>0}, \mathbb{1}_{|S\cap M|>0}\right).$$

The function f_P is then defined to be the OXS functions over unit-demand functions u_i, v_i, w_i.

The analysis. We consider a uniformly random function $f_P \in \mathcal{F}$. The main lemma is that elements in G and B are indistinguishable from one round of queries over any (potentially non-feasible) sets. The main observation to obtain the indistinguishability of G and B over large sets is that if set S contains sufficiently many masking elements M, then the marginal contribution of a good element to S is 1 due to w_i, which is equal to the marginal contribution of a bad element when S contains sufficiently many elements form $G \cup B$. If a set S contains no masking elements and at least $\log n$ elements from $G \cup B$, then good and bad elements have marginal contributions that are 2 and 1 respectively, which leads to the following.

Lemma 9. *Consider any algorithm $\mathcal{A}$ and a uniformly random function $f_P \in \mathcal{F}_r$. For any set S, $f_P(S)$ is independent of the partition of $G \cup B$ into G and B with probability $1 - n^{-\omega(1)}$.*

Proof. We split the analysis into two parts.

For small sample sizes: If we take a sample of size $t \le \log^3(n)$ the probability of seeing more than $\log(n)$ elements from either G or B is super-polynomially small:

$$\Pr(|(G\cup B)\cap S| > \log(n)) < \binom{t}{\log(n)}\left(\frac{|G\cup B|}{n}\right)^{\log(n)}$$

$$< \binom{\log^3(n)}{\log(n)}\left(\frac{2\log^6(n)}{n}\right)^{\log(n)} < \left(\frac{2\log^9(n)}{n}\right)^{\log(n)} < n^{-\log(\log(n))}$$

So w.h.p., the marginal contribution of good or bad elements is 2.

For large sample sizes: If we take a sample of size $t \ge \log^3(n)$, then the probability of seeing less than $\log^2(n)$ masking elements from set M is super-polynomially small:

$$\Pr(|M\cap S| < \log^2(n)) < \binom{t}{t-\log^2(n)}\left(\frac{|G\cup B|}{n}\right)^{t-\log^2(n)}$$

$$< \binom{n}{\log^2(n)}\left(\frac{\log^7(n)}{n}\right)^{\frac{\log^3(n)}{2}} < n^{\log^2(n)}\left(\frac{\log^7(n)}{n}\right)^{\frac{\log^3(n)}{2}}$$

$$= n^{-\log^3(n)/2+\log^2(n)+7\log^2(n)\log(\log(n))/2} < n^{-\log(n)}.$$

where the first inequality follows from bounding the probability that at least $t-\log^2(n)$ elements are in $G \cup B$ and the last inequality holds for large n. When the number of masking elements exceeds $\log^2(n)$, g_i and b_i are indistinguishable. □

E.2.1 Proof of Theorem 4

Proof. From Lemma 9, we may assume that w.h.p. g_i and b_i are indistinguishable. Since the algorithm may choose at most k elements from $G \cup B$ at random, we can bound the probability for selecting more than $\log(n)$ elements from G:

$$P(|G \cap S| > \log(n)) < \binom{k}{\log(n)} \left(\frac{|G|}{|G \cup B|}\right)^{\log(n)}$$

$$< \binom{\log^2(n)}{\log(n)} \left(\frac{\log^3(n)}{\log^6(n) + \log^3(n)}\right)^{\log(n)} < \left(\frac{\log^5(n)}{\log^6(n)}\right)^{\log(n)} = \log(n)^{-\log(n)} = n^{-\log(\log(n))}$$

So, for any $\delta > 0$ and large enough n, we get that with high probability $f(S) \leq \log^2(n) + 2\log(n)$ and $(2-\delta)f(S) < \texttt{OPT}$. □

E.3 Analysis for Hardness of $\tilde{o}(\log n)$ Rounds of Adaptivity

The construction. We consider the family of partitions $\mathcal{P}$ that consists of all partitions P such that the n elements are partitioned into $r+2$ disjoint sets, where $r = \frac{\log n}{4\log(\log n)} - 1$, as follows: the set G of good elements of size $k = \log^4 n$, sets B_i of bad elements of size $\log^{4(i+1)} n$ for $i \in [r]$, and the remaining elements are dummy elements comprising set M.

We construct OXS functions f_P which depend on a partition $P \in \mathcal{P}$, so the hard family of OXS functions is $\mathcal{F}_r = \{f_P : P \in \mathcal{P}\}$. Given a partition P, we define $\log^4 n + r\log^2 n$ unit-demand functions function g_i, for $i \in [\log^4 n]$, and $b_{i,j}$, for $i \in [r]$ and $j \in [\log^2 n]$, where

$$g_i(S) = \mathbb{1}_{|S \cap G| > 0} \quad \text{and} \quad b_j(S) = \mathbb{1}_{|S \cap B_i| > 0}.$$

The function f_P is then defined to be the OXS function over unit-demand functions $g_i, b_{i,j}$. Similar to the previous construction, we note that the set G has high value $f_P(G) = \log^4 n$ while the sets B_i have low value $f_P(B_i) = \log^2 n$ for all $i \in [r]$. We also note that for all i, $|B_i| = \log^4 n |B_{i-1}|$.

The analysis. We consider a uniformly random function $f_P \in \mathcal{F}_r$. The main lemma for this construction is, informally, that the algorithm can learn at most one part B_i of the partition P per round. More precisely, let $B^i = \bigcup_{i' \leq r-i} B_{i'}$. If the queries of the algorithms at round i are chosen independently of the partition of $B^i \cup G$ into $B_1, \ldots, B_{r-i}, G$, then the algorithm may learn B_{r-i}, but cannot learn any information about the partition of $B^{i+1} \cup G$ into $B_1, \ldots, B_{r-(i+1)}, G$. Additionally, if the algorithm cannot distinguish elements in B_1 and G, then, with high probability, the algorithm returns a solution that is at best a $1/\log n$ approximation to $f_P(G)$.

Lemma 10. *Consider a uniformly random function $f_P \in \mathcal{F}_r$. Let $B^i = \bigcup_{i' \leq r-i} B_{i'}$. For any collection of $\mathrm{poly}(n)$ queries $\{S_j\}_j$, each of size $\mathcal{O}(k)$, that are chosen independently of the partition of $B^i \cup G$ into $B_1, \ldots, B_{r-i}, G$, then, with probability smaller than $2^{-\log^2 n}$, for all S_j, the value $f_P(S_j)$ is independent of the partition of $B^{i+1} \cup G$ into $B_1, \ldots, B_{r-(i+1)}, G$.*

Proof. We show the claim by induction. Before the first round, we have no information. Assume we cannot distinguish between any element in B^{i-1} and G. Any algorithm samples at most ck elements from $B^i \cup G$ at random. The number of elements chosen from $B^i \cup G$ is bounded w.h.p.:

$$\Pr\left(|(G \cup B^i) \cap S| > \log^2(n)\right) < \binom{t}{\log^2(n)} \left(\frac{|G \cup B^i|}{|G \cup B^{i-1}|}\right)^{\log^2(n)}$$

$$< \binom{ck}{\log^2(n)} \left(\frac{1}{\log^3(n)}\right)^{\log^2(n)} \leq \left(\frac{c\log^4(n)}{\log^5(n)}\right)^{\log^2(n)} < 2^{-\log^2(n)}$$

Thus, w.h.p. the marginal contribution of B^i and G is equal to 1 so the sets are indistinguishable. □

Lemma 11. *Consider any algorithm $\mathcal{A}$ and a uniformly random function $f_P \in \mathcal{F}_r$. If for all queries S by $\mathcal{A}$, $f_P(S)$ is independent of the partition of $B_1 \cup G$ into B_1 and G, the probability of returning a set S such that $f_P(S) > \frac{f_P(G)}{\log n}$ is smaller than $2^{-\log^2 n}$.*

Proof. Since the algorithm may choose up to k elements from $G \cup B$ at random and with probability at least $1 - 2^{-\log^2(n)}$, G and B are indistinguishable, we can bound the probability for selecting more than d elements from G:

$$P(|G \cap S| > d) < \binom{k}{d} \left(\frac{|G|}{|G \cup B|}\right)^d < \binom{\log^4(n)}{d} \left(\frac{1}{\log^3(n)}\right)^d < \left(\frac{\log(n)}{d}\right)^{-d}$$

Setting $d = \log^2(n)$, we get that at most $\log^2(n)$ elements from G were selected. The contribution of all the other elements is bounded by $r \log^2(n)$ and thus, $f(S) < \log^3(n)$. Since $\texttt{OPT} = \log^4(n)$, the claim holds. □

The proof of Theorem 5 follows from combining both Lemma 10 and Lemma 11.

F Experimental Constructions

In this section, we discuss in detail the construction of synthetic graphs G3 and G4 and Twitter graphs.

Synthetic bipartite graphs. We generate randomized bipartite graphs with $m = 200$ players and $n = 200$ items, where the probability that an edge exists between a certain player and node is 0.25 for G3 and 0.75 for G4. This probability affects the number of neighbors the graph contains and the overall graph density. We select 75% of players and items to be "bad" nodes and the remaining node to be "good". We then insert an additional 50 item nodes into the graph as "masking" nodes and construct edges between these 50 nodes and 10 randomly chosen players and assign an edge weight of 1 to these edges. For the remaining edges, we assign low weights (generated uniformly from 0 and 0.1) to edges that contain any "bad" node and high weights (generated uniformly from 0.9 and 1) to edges that connect two "good" nodes. The edge weights were generated so that 60% of nodes had low weights and 40% of nodes had high weights.

Twitter graphs. For a particular hashtag, we first filter for 500 tweets that contain that particular hashtag. We then use the NLP library RAKE to extract keywords from each tweet. We discard all tweets with one or less keyword, to ensure the graph is of a certain density. For each keyword, we then calculate the frequency of that word in the English corpus on the Zipf scale, where the frequency is the base-10 logarithm of the number of times it appears per billion words. Scores generally fall between 0 and 8. To assign weights to each edge, we multiply the number of keywords of a tweet by the Zipf score of the keyword.

Number of rounds. For G1-G4 ($k = 10 : 100$ with leaps of 10) the number of rounds were: G1 Rounds = [18, 16, 14, 19, 17, 15, 20, 17, 15, 20] G2 Rounds = [3,3,4,5,6,7,8,9,16,16] G3 Rounds = [4, 6, 7, 8, 9, 10, 11, 13, 14, 16] G4 Rounds = [4, 6, 7, 8, 9, 10, 11, 13, 14, 16]. For $G5 - 8$ the number of rounds is approximately k with $\epsilon = 0.1$ as seen in the plots by the quality of TRIMMED GREEDY. Using $\epsilon = 0.3$ reduces the number of rounds dramatically without harming the results, and the plots are essentially identical up to **Trimmed Greedy** which now have the lowest performances. For example, on data set G7 with $k = 150$ and $\epsilon = 0.1$, GSAS uses 138 rounds while 42 rounds are sufficient for $\epsilon = 0.3$ with the same returned value.