

Submodular Max-SAT

Yossi Azar*

Iftah Gamzu[†]

Ran Roth[‡]

Abstract

We introduce the submodular Max-SAT problem. This problem is a natural generalization of the classical Max-SAT problem in which the additive objective function is replaced by a submodular one. We develop a randomized linear-time $2/3$ -approximation algorithm for the problem. Our algorithm is applicable even for the online variant of the problem. We also establish hardness results for both the online and offline settings. Notably, for the online setting, the hardness result proves that our algorithm is best possible, while for the offline setting, the hardness result establishes a computational separation between the classical Max-SAT and the submodular Max-SAT.

*Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. Email: azar@tau.ac.il.

[†]Microsoft R&D Center, Herzliya 46725, Israel. Email: iftah.gamzu@cs.tau.ac.il.

[‡]Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. Email: ranroth@tau.ac.il.

1 Introduction

Max-SAT is one of the most fundamental combinatorial optimization problems in computer science. As input for this problem, we are given a set of boolean variables and a collection weighted CNF clauses. The objective is to find a truth assignment for the variables which maximizes the sum of weights of satisfied clauses. In recent years, there has been a surge of interest in understanding the limits of tractability of optimization problems in which the classic additive objective function was replaced by a submodular one. Submodularity arises naturally in many practical scenarios, most notably in economics due to the property of diminishing returns. In consequence, it seems natural to study the submodular Max-SAT problem. In this variant, the value of a given assignment is the value that a monotone submodular function gives to the set of satisfied clauses. Note that in the special case in which this function is additive then one obtains the classical Max-SAT.

In this paper, we concentrate on developing fast approximation algorithms for submodular Max-SAT. In particular, we are interested in linear time algorithms. Furthermore, we study the online version of submodular Max-SAT. In this variant, the variables arrive one by one. Once a variable arrives, it declares the clauses in which it (or its negation) participates, and then, the algorithm needs to make an irrevocable decision regarding its assignment. Clearly, the algorithm does not have any information about the variables that will arrive in the future and the clauses in which they will appear.

Our contribution. Our results can be briefly described as follows:

- We develop a $2/3$ -approximation algorithm for the submodular Max-SAT problem. Our algorithm is combinatorial, randomized, simple to implement, and runs in linear time. In fact, our algorithm does a single pass over the input (in some arbitrary order), and thus, it is also applicable in the online setting.
- We establish that no online algorithm (deterministic or randomized) can attain a competitive ratio better than $2/3$ for the online version of the classical Max-SAT problem. This result clearly holds also for the more general online submodular Max-SAT problem. Our result implies that the analysis of our randomized algorithm is tight, and that it is best possible in the online setting.
- We prove an information-theoretic inapproximability result of $3/4$ for the offline variant of the submodular Max-SAT problem. This result is based on an observation regarding the equivalence of submodular Max-SAT and the problem of maximizing a monotone submodular function under a so-called binary partition matroid. The hardness result then follows by observing that problem of combinatorial auctions with two submodular bidders is a special instance of our problem in which the monotone submodular function has an additional constraint on its structure. For this problem, Mirrokni, Schapira and Vondrák [21] established a hardness bound of $3/4$. We also provide an alternative proof which may be interesting on its own right.

An interesting consequence of our results is an identification of a computational separation between the classical Max-SAT and its submodular counterpart. In particular, this separation is obtained by noticing that the classical Max-SAT can be approximated to within a factor strictly better than $3/4$ [3].

Related work. The classical Max-SAT problem has been given significant attention in the past (see, e.g., [29, 14, 13, 2, 18, 4]). It is known to be NP-hard to approximate within a factor better than $7/8 = 0.875$ [16, 22]. The best algorithm for the problem, presented by Asano [3], achieves a provable approximation guarantee of 0.7877 and a conjectured approximation ratio of 0.8353 . Specifically, the latter ratio relies on a conjecture of Zwick [30]. The problem of finding combinatorial and possibly online algorithms to Max-SAT has also gained notable attention. In a pioneer work in approximation algorithms, Johnson [17] suggested a greedy algorithm that uses modified weights. This algorithm was proved to achieve a $2/3$ -approximation by Chen,

Friesen and Zheng [7] (see also [10]). Very recently, a randomized combinatorial algorithm achieving a $3/4$ -approximation was presented by Poloczek and Schnitger [25]. Their result also holds for the online variant of Max-SAT. Note that this result does not contradict our online hardness result as they assume that the length of the clauses is known in advance. Another online variant of Max-SAT has been studied by Coppersmith et al. [8]. We remark that this variant is different than ours in several ways. One example of this difference is that clauses rather than the variables appear in an online fashion.

The submodular Max-SAT problem is equivalent to the problem of maximizing a monotone submodular function under a particular matroid constraint we name a binary partition matroid. This result is established in Subsection 3.1. There has been a long line of research on maximizing monotone submodular functions subject to matroid constraints. Arguably, the simplest scenario is maximizing a submodular function under a uniform matroid. This problem admits a tight approximation of $1 - 1/e \approx 0.632$ [24, 23, 11]. Recently, Calinescu et al. [5] utilized a continuous relaxation approach to achieve a $(1 - 1/e)$ -approximation for monotone submodular maximization under any matroid. In particular, due to the equivalency stated before, this algorithm can be applied to (the offline variant of) our problem. There has also been an ever-growing research on maximizing a monotone submodular function under additional constraints [28, 26, 15, 19, 20, 6].

2 Preliminaries

Submodular functions. A set function $f : 2^X \rightarrow \mathbb{R}$ is called *submodular* if

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T),$$

for all $S, T \subseteq X$, and it is called *monotone* if $f(S) \leq f(T)$ whenever $S \subseteq T$. We remark that without loss of generality we can restrict attention to normalized functions, i.e., $f(\emptyset) = 0$. An alternative definition of submodularity is through the property of decreasing marginal values. Given a function f and a set $S \subseteq X$, the function f_S is defined by $f_S(a) = f(S \cup \{a\}) - f(S)$. The value $f_S(a)$ is called the marginal value of element a to the set S . The *decreasing marginal values* property requires that $f_S(a)$ be a non-increasing function of S for every fixed a . Formally, it requires that $f_S(a) \geq f_T(a)$, for all $S \subseteq T$ and $a \in X \setminus T$. Note that the amount of information necessary to convey an arbitrary submodular function may be exponential. Hence, we assume a value oracle access to the function. A *value oracle* for f allows us to query about the value of $f(S)$ for any set S .

Submodular Max-SAT. An input instance of submodular Max-SAT consists of a set $V = \{v_1, \dots, v_n\}$ of boolean variables, and a collection $C = \{c_1, \dots, c_m\}$ of clauses, where each clause is a disjunction of literals over the variables in V . Let $f : C \rightarrow \mathbb{R}_+$ be a monotone submodular function over the clauses. Given an assignment $\eta : V \rightarrow \{\text{True}, \text{False}\}$, we denote by $C(\eta) \subseteq C$ the subset of clauses satisfied by η . The objective is to find an assignment η that maximizes $f(C(\eta))$ over all possible assignments. We note that the classical Max-SAT problem is obtained as a special case when f is an additive function. An *additive* function can be represented as a sum of weights.

Online Max-SAT. In the online Max-SAT problem, we are given the same input components. However, in this variant, the variables are introduced in an online fashion, namely, one by one. Whenever a variable v_i is introduced, it arrives with two lists of clauses. The first list consists of all the clauses that contain v_i , and the other list consists of all the clauses that contain its negation $\neg v_i$. Any online algorithm must make an irrevocable decision regarding the assignment of v_i without prior knowledge about additional variables that will arrive in the future.

Partition matroids. A *matroid* is a pair $\mathcal{M} = (X, \mathcal{I})$, such that X is a ground set of elements, and $\mathcal{I} \subseteq 2^X$ is a family of independent subsets, containing the empty set and satisfying two additional properties:

1. an *inheritance property*: if $S \in \mathcal{I}$ and $T \subseteq S$ then $T \in \mathcal{I}$, and
2. an *exchange property*: if $S, T \in \mathcal{I}$ and $|T| \leq |S|$ then there is $a \in S$ such that $T \cup \{a\} \in \mathcal{I}$.

A *partition matroid* defines a partition $\bigcup_{t=1}^m P_t = X$ of the ground set elements. Then, the family of independent subsets are defined as $\mathcal{I} = \{S \subseteq X : |S \cap P_t| \leq k_t \ \forall t \in [m]\}$, where each $k_t \in \mathbb{N}$ designates a cardinality bound of the corresponding partition subset. When all the partition subsets have a cardinality of two, we obtain a *binary partition matroid*. Note that without loss of generality we can assume that every independent subset consists of at most one element from each partition subset. More generally, when all the partition subsets have the same cardinality ℓ , and all cardinality bounds are k , we call the matroid a (k, ℓ) -*partition matroid*.

3 A Linear Time Approximation Algorithm

In this section, we describe a fast randomized approximation algorithm for the submodular Max-SAT problem. Our algorithm can be implemented to run in linear time, and it achieves an expected approximation guarantee of $2/3$. In fact, our algorithm can be applied when the input is introduced in an online fashion, and thus, it may be employed for the online variant of submodular Max-SAT. Prior to presenting the algorithm, we demonstrate that submodular Max-SAT is equivalent to the problem of maximizing a monotone submodular function subject to a binary partition matroid, formally defined below. We find this observation useful on its own merit as it connects our study with the long line of research on maximizing a monotone submodular function subject to a matroid constraint.

3.1 Submodular maximization under a binary partition matroid

An instance of the problem of maximizing a monotone submodular function subject to a binary partition matroid consists of a ground set $X = \{a_1, b_1, a_2, b_2, \dots, a_m, b_m\}$ of $n = 2m$ elements, a monotone submodular function $f : 2^X \rightarrow \mathbb{R}_+$, and a binary partition matroid $\mathcal{M} = (X, \mathcal{I})$ where each partition subset $P_t = \{a_t, b_t\}$. In particular, the family of independent subsets of this matroid is $\mathcal{I} = \{S \subseteq X : |S \cap P_t| \leq 1 \ \forall t \in [m]\}$. The objective is to maximize $f(S)$ subject to the constraint that $S \in \mathcal{I}$, that is, S consists of at most one element from each partition subset.

Theorem 3.1. *Submodular Max-SAT is equivalent to the problem of maximizing a monotone submodular function under a binary partition matroid.*

Proof. In what follows, we prove that any input instance of one problem can be translated to an input instance of the other problem in polynomial time. Note that each of these reductions maintains the solution value, that is, any solution to one problem can be translated to a solution to the other problem with the same value. Assume that we are given an input instance of submodular Max-SAT where we would like to maximize a monotone submodular function $g : 2^C \rightarrow \mathbb{R}_+$ for subsets of clauses having no contradiction. Let Lit be the set of literals of V , and let $\alpha : 2^{Lit} \rightarrow 2^C$ be the function that given a collection of literals returns the set of clauses that contain at least one of these literals. Formally,

$$\alpha(L) = \{c \in C : \text{there is } \ell \in L \text{ such that } \ell \text{ appears in } c\}$$

Having these definitions in mind, we define an instance of monotone submodular maximization under a binary partition matroid as follows: each partition subset corresponds to the two possible assignments (i.e., two literals) for a variable, and $f : 2^{Lit} \rightarrow \mathbb{R}_+$ is defined as $f(L) = g(\alpha(L))$. It is not difficult to verify

that maximizing f on this binary partition matroid is equivalent to maximizing g under the original Max-SAT constraints. Hence, we focus on proving that f is indeed monotone and submodular. Notice that

$$\begin{aligned}
f(S) + f(T) &= g(\alpha(S)) + g(\alpha(T)) \\
&\geq g(\alpha(S) \cup \alpha(T)) + g(\alpha(S) \cap \alpha(T)) \\
&\geq g(\alpha(S \cup T)) + g(\alpha(S \cap T)) \\
&= f(S \cup T) + f(S \cap T),
\end{aligned}$$

where the first inequality is due to the submodularity of g , and the last inequality follows by the monotonicity of g combined with the facts that $\alpha(S \cup T) = \alpha(S) \cup \alpha(T)$ and $\alpha(S \cap T) \subseteq \alpha(S) \cap \alpha(T)$. Consequently, f is submodular. In addition, it is easy to validate that f is monotone since g is monotone and $\alpha(L) \subseteq \alpha(L \cup \{\ell\})$, for any $\ell \in L$.

We now assume that we are given an input instance of monotone submodular maximization under a binary partition matroid. We define an instance of submodular Max-SAT as follows: we create a boolean variable v_t for every partition subset P_t , and arbitrarily associate the literal v_t with a_t and the literal $\neg v_t$ with b_t . Furthermore, we define the set of clauses to be all the singleton clauses on the literals, that is, $C = \{v_1, \neg v_1, v_2, \neg v_2, \dots, v_n, \neg v_n\}$. Notice that this induces a one-to-one correspondence between the clauses of C and the elements of X . Accordingly, we let $\beta : 2^C \rightarrow 2^X$ be the function that takes any subset of clauses and returns its corresponding set of elements under the one-to-one correspondence. Finally, we define $g(C) = f(\beta(C))$. One can easily verify that the function g is monotone and submodular, and that maximizing g under the Max-SAT constraints is equivalent to maximizing f on the original binary partition matroid. ■

3.2 The algorithm

In the following, we describe the approximation algorithm for the problem of maximizing a monotone submodular function under a binary partition matroid. As a result of the equivalency described in Subsection 3.1, we obtain an approximation algorithm for submodular Max-SAT. Our proportional select algorithm, formally described below, considers the partition subsets of the matroid in an arbitrary order, and selects one element from each partition subset. Unlike a greedy algorithm, which chooses a partition element with maximal marginal contribution, our algorithm randomly selects one of the two elements in proportion to their marginal contribution. This strategy turns out to be significantly better than using the natural greedy selection rule. We note that our algorithm has similarities with the algorithm of Dobzinski and Schapira [9] for the problem of combinatorial auctions with two submodular bidders. Recall that the latter problem is a special instance of our problem in which the monotone submodular function has an additional constraint on its structure. Also recall that $f_S(a)$ is the incremental marginal value of element a to the set S .

Analysis. In what follows, we prove that the proportional select algorithm attains an approximation ratio of $2/3$. For ease of presentation, we use $+$ and $-$ to denote set union and set difference, respectively. We begin by introducing the definition of an optimal solution O_A constrained by a given (feasible) solution A .

Definition 3.2. Let $A \subseteq X$ be an independent set of a matroid \mathcal{M} . The set $O_A \subseteq X$ is defined as an optimal solution to the problem of maximizing f under \mathcal{M} that satisfies $A \subseteq O_A$. Formally,

$$O_A = \operatorname{argmax}_{T \in \mathcal{I}, A \subseteq T} f(T).$$

Moreover, we let $\operatorname{OPT}_A = f(O_A)$ be the value that f assigns the set O_A . Note that the value of the optimal solution that maximizes f under \mathcal{M} is $\operatorname{OPT} = \operatorname{OPT}_\emptyset$.

Algorithm 1 Proportional Select

Input: A monotone submodular function $f : 2^X \rightarrow \mathbb{R}_+$ and a binary matroid \mathcal{M}

Output: A set $S \subseteq X$ approximating the maximum of f under \mathcal{M}

- 1: $S_0 \leftarrow \emptyset$
 - 2: **for** $t \leftarrow 1$ to m **do**
 - 3: $w_t \leftarrow f_{S_{t-1}}(a_t) + f_{S_{t-1}}(b_t)$
 - 4: $p_{a_t} \leftarrow f_{S_{t-1}}(a_t)/w_t$
 - 5: $p_{b_t} \leftarrow f_{S_{t-1}}(b_t)/w_t$
 - 6: Pick s_t at random from $\{a_t, b_t\}$ with respective probabilities (p_{a_t}, p_{b_t})
 - 7: $S_t \leftarrow S_{t-1} \cup \{s_t\}$
 - 8: **end for**
 - 9: $S \leftarrow S_m$
-

We turn to bound the loss of O_A when it is constrained to select an element that is not part of it.

Lemma 3.3. *Let $A \subseteq X$ be an independent set of a matroid \mathcal{M} such that $A \cap P_t = \emptyset$. Moreover, let x_t be the element of P_t that belongs to O_A and y_t the element of P_t that does not appear in O_A . Then,*

$$\text{OPT}_A - \text{OPT}_{A+y_t} \leq f_A(x_t).$$

Proof. One can easily verify that

$$\begin{aligned} \text{OPT}_A = f(O_A) &\leq f(O_A + y_t) \\ &= f(O_A + y_t) - f(O_A + y_t - x_t) + f(O_A + y_t - x_t) \\ &= f_{O_A+y_t-x_t}(x_t) + f(O_A + y_t - x_t), \end{aligned}$$

where the inequality results from the monotonicity of f . Notice that $A \subseteq O_A + y_t - x_t$, and therefore, $f_{O_A+y_t-x_t}(x_t) \leq f_A(x_t)$ by the submodularity of f . Furthermore,

$$f(O_A - x_t + y_t) \leq f(O_A + y_t) = \text{OPT}_{A+y_t},$$

where the inequality follows as O_{A+y_t} is optimal with respect to $A + y_t \subseteq O_A - x_t + y_t$. In consequence, we obtain that $\text{OPT}_A \leq f_A(x_t) + \text{OPT}_{A+y_t}$. ■

We now analyze the performance of our algorithm. For this purpose, we define the loss of the algorithm at step t of the main loop as $L_t = \text{OPT}_{S_{t-1}} - \text{OPT}_{S_t}$. The following observation makes a connection between the sum of losses along the steps of the algorithm and the difference between the value of the optimal solution, OPT , and the solution of our algorithm, $f(S)$.

Observation 3.4. $\sum_{t=1}^m L_t = \text{OPT} - f(S)$

Proof. Notice that

$$\sum_{t=1}^m L_t = \sum_{t=1}^m (\text{OPT}_{S_{t-1}} - \text{OPT}_{S_t}) = \text{OPT}_{S_0} - \text{OPT}_{S_m} = \text{OPT}_{\emptyset} - f(S_m) = \text{OPT} - f(S)$$

■

We are ready to prove the main theorem of this section, stating that the expected value of the solution that our algorithm generates is at least $2/3$ of the value of the optimal solution.

Theorem 3.5. $\mathbb{E}[f(S)] \geq \frac{2}{3} \text{OPT}$

Proof. Let us focus on step t of the algorithm in which one of the elements of the partition subset P_t is selected. Note that only one of the elements of P_t belongs to $O_{S_{t-1}}$. Let us denote that element by x_t and the element of P_t that is not in $O_{S_{t-1}}$ by y_t .

We begin by bounding the expected loss of the algorithm at step t , given the set of elements selected up to step $t - 1$. Recall that s_t is the element selected at step t of the algorithm.

$$\begin{aligned} \mathbb{E}[L_t | S_{t-1}] &= \Pr[s_t = x_t | S_{t-1}] \cdot (\text{OPT}_{S_{t-1}} - \text{OPT}_{S_{t-1}+x_t}) + \\ &\quad \Pr[s_t = y_t | S_{t-1}] \cdot (\text{OPT}_{S_{t-1}} - \text{OPT}_{S_{t-1}+y_t}) \\ &= \Pr[s_t = x_t | S_{t-1}] \cdot 0 + \Pr[s_t = y_t | S_{t-1}] \cdot (\text{OPT}_{S_{t-1}} - \text{OPT}_{S_{t-1}+y_t}) \\ &\leq \Pr[s_t = y_t | S_{t-1}] \cdot f_{S_{t-1}}(x_t) \\ &= \frac{f_{S_{t-1}}(y_t) f_{S_{t-1}}(x_t)}{f_{S_{t-1}}(x_t) + f_{S_{t-1}}(y_t)}, \end{aligned}$$

where the inequality is due to Lemma 3.3, and the last equality is attained by recalling that the algorithm selects y_t with probability $f_{S_{t-1}}(y_t)/(f_{S_{t-1}}(x_t) + f_{S_{t-1}}(y_t))$. We now turn to calculate the expected gain of the algorithm in step t , given the set of elements selected up to step $t - 1$.

$$\begin{aligned} \mathbb{E}[f_{S_{t-1}}(s_t) | S_{t-1}] &= \Pr[s_t = x_t | S_{t-1}] \cdot f_{S_{t-1}}(x_t) + \Pr[s_t = y_t | S_{t-1}] \cdot f_{S_{t-1}}(y_t) \\ &= \frac{f_{S_{t-1}}(x_t)}{f_{S_{t-1}}(x_t) + f_{S_{t-1}}(y_t)} \cdot f_{S_{t-1}}(x_t) + \frac{f_{S_{t-1}}(y_t)}{f_{S_{t-1}}(x_t) + f_{S_{t-1}}(y_t)} \cdot f_{S_{t-1}}(y_t) \\ &= \frac{f_{S_{t-1}}(x_t)^2 + f_{S_{t-1}}(y_t)^2}{f_{S_{t-1}}(x_t) + f_{S_{t-1}}(y_t)} \end{aligned}$$

This implies that the expected loss to gain ratio is

$$\frac{\mathbb{E}[L_t | S_{t-1}]}{\mathbb{E}[f_{S_{t-1}}(s_t) | S_{t-1}]} \leq \frac{f_{S_{t-1}}(x_t) f_{S_{t-1}}(y_t)}{f_{S_{t-1}}(x_t)^2 + f_{S_{t-1}}(y_t)^2} \leq \frac{1}{2},$$

where the last inequality holds since $2ab \leq a^2 + b^2$, for any $a, b \in \mathbb{R}$. We can now bound the expected loss of the algorithm at step t as follows

$$\begin{aligned} \mathbb{E}[L_t] &= \sum_{S_{t-1} \subseteq X} \mathbb{E}[L_t | S_{t-1}] \cdot \Pr \left[\begin{array}{l} \text{the algorithm selects} \\ S_{t-1} \text{ up to step } t-1 \end{array} \right] \\ &\leq \frac{1}{2} \sum_{S_{t-1} \subseteq X} \mathbb{E}[f_{S_{t-1}}(s_t) | S_{t-1}] \cdot \Pr \left[\begin{array}{l} \text{the algorithm selects} \\ S_{t-1} \text{ up to step } t-1 \end{array} \right] \\ &= \frac{1}{2} \mathbb{E}[f_{S_{t-1}}(s_t)] \end{aligned}$$

Consequently, we get that

$$\text{OPT} - \mathbb{E}[f(S)] = \mathbb{E} \left[\sum_{t=1}^m L_t \right] = \sum_{t=1}^m \mathbb{E}[L_t] \leq \frac{1}{2} \sum_{t=1}^m \mathbb{E}[f_{S_{t-1}}(s_t)] = \frac{1}{2} \mathbb{E} \left[\sum_{t=1}^m f_{S_{t-1}}(s_t) \right] = \frac{1}{2} \mathbb{E}[f(S)],$$

where the first equality follows from Observation 3.4. Thus, $\mathbb{E}[f(S)] \geq 2/3 \cdot \text{OPT}$. \blacksquare

The following corollary summarizes the properties of our algorithm.

Corollary 3.6. *Algorithm proportional select runs in linear time and achieves an expected approximation ratio of $2/3$ for the submodular Max-SAT problem.*

One can easily verify that the order in which our algorithm considers the partition subsets is not really important. This implies that the algorithm may be applied in case the partition subsets are introduced in an online fashion. In this setting, the two elements of a partition subset arrive in each step, and the algorithm needs to make an irrevocable decision which element to select to its solution (without prior knowledge about partition subsets that will arrive in the future). Using the equivalency described in Subsection 3.1, one can observe that the algorithm may be applied to the online Max-SAT problem. Specifically, it is not hard to validate that the two elements of each partition subset can correspond to the two possible assignments for a variable. In consequence, we obtain the following corollary.

Corollary 3.7. *Algorithm proportional select achieves an expected competitive ratio of $2/3$ for the online submodular Max-SAT problem.*

4 A Hardness Bound for the Online Version

In this section, we demonstrate that no *randomized* online algorithm can guarantee a competitive ratio better than $2/3$ for the online version of Max-SAT. We emphasize that this hardness bound also holds for the online variant of the classical Max-SAT. This implies that the analysis of our algorithm from Section 3 is tight, and that this algorithm is optimal for the online setting of both the submodular and classical versions of Max-SAT. We also note that one can easily prove a hardness bound of $1/2$ for any *deterministic* online algorithm. Interestingly, this establishes that randomization provably helps in our online setting.

Theorem 4.1. *No online algorithm can attain a competitive ratio better than $2/3$ for online Max-SAT.*

Proof. In what follows, we present a distribution over inputs such that any deterministic online algorithm, given an input drawn from this distribution, achieves an expected competitive ratio of at most $2/3 + \epsilon$ for any fixed $\epsilon > 0$. By Yao's principle, it then follows that any randomized online algorithm cannot achieve an expected competitive ratio better than $2/3$.

Let $V = \{v_1, \dots, v_n\}$ be a set of boolean variables in our input. The variables are introduced according to their order, namely, v_1, v_2 , and so on. We have $m = 2^n$ clauses with identical weights. Whenever a variable v_i is introduced, the algorithm is given the subsets S_i and T_i . Specifically, S_i consists of all the clauses in which v_i appears, and T_i consists of all the clauses in which $\neg v_i$ appears. We now formally define the sets S_i, T_i that correspond to each variable v_i . At the first step, when v_1 is presented to the algorithm, $S_1 = \{c_1, \dots, c_{m/2}\}$ and $T_1 = \{c_{m/2+1}, \dots, c_m\}$, that is, the clauses $c_1, \dots, c_{m/2}$ contain v_1 and the rest of the clauses contain $\neg v_1$. At the next step, when v_2 is presented to the algorithm, one of the subsets S_1, T_1 is randomly chosen. Let C_1 be the chosen subset, and note that $|C_1| = m/2$. The set S_2 is defined to consist of the first $m/4$ clauses in C_1 , while T_2 consists of the remaining $m/4$ clauses in C_1 . The same process continues with all the remaining variables. In particular, at step i , when v_i is presented to the algorithm, one of the subsets S_{i-1}, T_{i-1} is randomly chosen as C_{i-1} . Then, the set S_i is defined to consist of the first $|C_{i-1}|/2$ clauses of C_{i-1} , and T_i consists of the remaining $|C_{i-1}|/2$ clauses of C_{i-1} . A concrete construction of clauses is presented in Figure 1.

Notice that each chosen set C_i holds the *active* clauses, namely, clauses that may be affixed with additional variables later in the randomized construction. As a result, by choosing $v_i = \text{False}$ whenever $C_i = S_i$ and $v_i = \text{True}$ whenever $C_i = T_i$, one can obtain an optimal assignment which satisfies all clauses but one. Thus,

$$\begin{aligned}
c_1 &= v_1, & c_5 &= \neg v_1 \vee v_2 \\
c_2 &= v_1, & c_6 &= \neg v_1 \vee v_2 \\
c_3 &= v_1, & c_7 &= \neg v_1 \vee \neg v_2 \vee v_3 \\
c_4 &= v_1, & c_8 &= \neg v_1 \vee \neg v_2 \vee \neg v_3
\end{aligned}$$

Figure 1: A construction of clauses for $n = 3$ under the assumption that always $C_i = T_i$.

we have $\text{OPT} = m - 1$. We turn to calculate the expected number of clauses satisfied by any online algorithm. Let us focus on step i . One can validate that choosing $v_i = \text{True}$ if $C_i = S_i$ or $v_i = \text{False}$ if $C_i = T_i$ results in an assignment that cannot satisfy more than $m \cdot (1 - 2^{-i})$ of the clauses. However, since the assignment of v_i is done without any information about the random choice of C_i it follows that any online algorithm makes a wrong choice with probability $1/2$. In consequence, it generates an assignment that cannot satisfy more than $m \cdot (1 - 2^{-i})$ of the clauses. Notice that the probability that the algorithm makes its first wrong choice at step i is 2^{-i} . Hence,

$$\begin{aligned}
\mathbb{E} \left[\begin{array}{c} \text{number of} \\ \text{satisfied clauses} \end{array} \right] &= 2^{-n} \cdot (m - 1) + \sum_{i=1}^n 2^{-i} \cdot m \cdot (1 - 2^{-i}) \\
&\leq m \cdot \left[2^{-n} + \sum_{i=1}^n 2^{-i} - \sum_{i=1}^n 4^{-i} \right] \\
&= m \cdot \left[1 - \left(\frac{1}{3} - \frac{1}{3 \cdot 4^n} \right) \right] \\
&= \frac{2}{3}m + \frac{m}{3 \cdot 4^n} \leq \frac{2}{3}m + \frac{1}{m},
\end{aligned}$$

where the first equality holds since an algorithm cannot make a wrong choice in the last step. As m tends to infinity, the expected competitive ratio approaches $2/3$, as required. We note that although the number of clauses in our construction is exponential in n , our hardness result also holds if the number of clauses is required to be polynomial in n . In such case, one can construct an instance as before with $\log m$ variables and then add dummy variables which do not appear in any clause. ■

5 A Hardness Bound for the Offline Version

In this section, we show that any algorithm, possibly having unlimited computational power, cannot achieve an approximation ratio better than $3/4$ for monotone submodular maximization under a binary partition matroid without asking exponentially-many oracle queries to the function. As a result of the equivalency described in Subsection 3.1, we obtain the same hardness result for submodular Max-SAT. We note that Mirrokni, Schapira and Vondrák [21] established a hardness bound of $3/4$ for the problem of combinatorial auctions with two submodular bidders. Since the latter problem is a special instance of our problem, their bound is also applicable to our setting. Here, we provide an alternative proof, which may be interesting on its own right. Our result is based on an adaptation of the approach of Vondrák [27], but with an additional novel technique. The approach of Vondrák employs a matroid refinement procedure, which does not preserve the structure of the underlying matroid, and thus, cannot be used in our case, where each partition subset has a bounded cardinality. Specifically, using this procedure, we would have received a hardness bound for a $(k, 2k)$ -partition

matroid where k can be very large. To overcome this shortcoming, we develop a new technique that we name *matroid blowup*. This technique preserves the structural properties of the underlying partition matroid and maintains all the properties required by the original approach. We begin by defining several concepts that will be used for the exposition of our technique and its analysis, and stating a lemma proved in [27].

Definition 5.1. Let $f : 2^X \rightarrow \mathbb{R}$ be a set function. One can naturally represent a subset $S \subseteq X$ as an indicator vector $x \in \{0, 1\}^X$ in which the entry x_a indicates whether $a \in S$ or not. Calinescu et al. [5] proposed one useful way to extend f to work on vectors in $[0, 1]^X$ by viewing x_a as the probability that $a \in S$. Formally, the *multilinear extension* $F : [0, 1]^X \rightarrow \mathbb{R}$ of f is defined as

$$F(x) = \mathbb{E}[f(\hat{x})] = \sum_{S \subseteq X} f(S) \prod_{a \in S} x_a \prod_{b \notin S} (1 - x_b).$$

Here, \hat{x} is a random variable in $\{0, 1\}^X$, where each entry a is rounded to 1 with probability x_a , or 0 otherwise.

Definition 5.2. The *polytope* $\mathcal{P}(\mathcal{M})$ of a matroid $\mathcal{M} = (X, \mathcal{I})$ is defined to be the set of all vectors in $[0, 1]^X$ that can be represented as a convex combination of indicator vectors of independent subsets of \mathcal{I} . More precisely, a vector $x \in [0, 1]^X$ belongs to $\mathcal{P}(\mathcal{M})$ if $x = \sum_i \lambda_i x(S_i)$, where $x(S_i)$ is the indicator vector that corresponds to an independent subset $S_i \in \mathcal{I}$, each λ_i is non-negative, and $\sum_i \lambda_i = 1$.

Definition 5.3. Let X be a ground set of elements, and let $\mathcal{M} = (X, \mathcal{I})$ be a matroid. Moreover, let $f : 2^X \rightarrow \mathbb{R}$ be a set function, and let $F : [0, 1]^X \rightarrow \mathbb{R}$ be its multilinear extension. Finally, let \mathcal{G} be a group of permutations on X . We define the following:

- The instance $\max\{f(S) : S \in \mathcal{I}\}$ is *invariant under* \mathcal{G} if for any $\sigma \in \mathcal{G}$ and $S \subseteq X$ it holds that $f(S) = f(\sigma(S))$ and $S \in \mathcal{I} \Leftrightarrow \sigma(S) \in \mathcal{I}$.
- Let $x \in [0, 1]^X$. The *symmetrized version* of x is $\bar{x} = \mathbb{E}_{\sigma \in \mathcal{G}}[\sigma(x)]$. Here, $\sigma(x)$ indicates the permutation of the components of x according to σ .
- Recall that $\mathcal{P}(\mathcal{M}) \subseteq [0, 1]^X$ is the polytope of the matroid \mathcal{M} . Let the optimum value of F constrained to $\mathcal{P}(\mathcal{M})$ be $\text{OPT} = \max\{F(x) : x \in \mathcal{P}(\mathcal{M})\}$, and similarly, let the symmetrized optimum of F be $\overline{\text{OPT}} = \max\{F(\bar{x}) : x \in \mathcal{P}(\mathcal{M})\}$. The *symmetry gap* of the underlying instance is defined as $\gamma = \overline{\text{OPT}}/\text{OPT}$.

Lemma 5.4. ([27, Lemma 3.2]) *Consider an instance $\max\{f(S) : S \in \mathcal{I}\}$ invariant under \mathcal{G} , where f is monotone and submodular, and fix any $\epsilon > 0$. There is $\delta > 0$ and functions $\hat{F}, \hat{G} : [0, 1]^X \rightarrow \mathbb{R}$ satisfying*

- Whenever $\|x - \bar{x}\|^2 \leq \delta$, it holds that $\hat{F}(x) = \hat{G}(x)$ and the value depends only on \bar{x} .
- $\max\{\hat{F}(x) : x \in \mathcal{P}(\mathcal{M})\} \geq \text{OPT}$.
- $\max\{\hat{G}(x) : x \in \mathcal{P}(\mathcal{M})\} \leq (1 + \epsilon)\overline{\text{OPT}}$.
- $\frac{\partial \hat{F}}{\partial x_i} \geq 0$ and $\frac{\partial \hat{G}}{\partial x_i} \geq 0$.
- $\frac{\partial^2 \hat{F}}{\partial x_i \partial x_j} \leq 0$ and $\frac{\partial^2 \hat{G}}{\partial x_i \partial x_j} \leq 0$.

We now define the notion of a partition matroid blowup. This definition is analogous to the notion of matroid refinement [27], but more restrictive. Specifically, this blowup procedure preserves the structural properties of the underlying partition matroid whereas a matroid refinement does not.

Definition 5.5. Let $X = \{a_1, \dots, a_n\}$ be a ground set of elements, and let $\mathcal{M} = (X, \mathcal{I})$ be a (k, ℓ) -partition matroid with partition subsets P_1, \dots, P_m . Let $r \in \mathbb{N}_+$, and let $R = [r]$. The r -blowup of \mathcal{M} , denoted by $\tilde{\mathcal{M}}$, is a (k, ℓ) -partition matroid defined on the ground set $R \times X$ with the partition subsets $P_{i,t} = \{i\} \times P_t$, for all $i \in R, t \in [m]$. We call the index i of $P_{i,t}$ its layer.

In other words, in order to obtain $\tilde{\mathcal{M}}$, we create r layers, each of which consists of a duplicate of the partition structure of the original matroid. Notice that the result of the blowup is still a (k, ℓ) -partition matroid, but it has r times as many partition subsets as the original matroid had. Figure 2 provides a schematic illustration of such a r -blowup procedure.

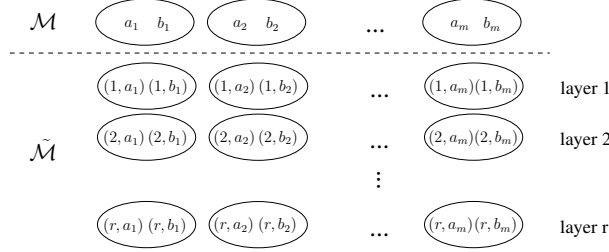


Figure 2: A r -blowup of a binary partition matroid.

Definition 5.6. Let $S \subseteq R \times X$. Moreover, for each $i \in R$, let $\sigma^{(i)} \in \mathcal{G}$ be some permutation of X . We define the vector $\xi(S) \in [0, 1]^X$ by

$$\xi_j(S) = \frac{1}{r} \left| \{i \in R : (i, \sigma^{(i)}(a_j)) \in S\} \right|$$

Notice that if $\sigma^{(i)}$ is the identity permutation for all layers i then $\xi_j(S)$ is the number of layers in which the copies corresponding to a_j belong to S divided by the total number of layers. Also note that applying a different choice of permutations $\{\sigma^{(i)}\}_{i=1}^r$ is equivalent to first shuffling the labels of the elements in each layer according to its permutation and then performing the same measurement.

We continue by citing the following lemma stated in [27]. We remark that this lemma was originally stated with respect to identity permutations and not an arbitrary set $\{\sigma^{(i)}\}_{i=1}^r$ of permutations. Nevertheless, it is easy to verify that the conclusion still applies since the effect of each $\sigma^{(i)}$ is just a reshuffling.

Lemma 5.7. ([27, Lemma 3.1]) *Let $H : [0, 1]^X \rightarrow \mathbb{R}$ be a function with continuous first partial derivatives everywhere, and second partial derivatives almost everywhere. Assume that for any i , $\frac{\partial H}{\partial x_i} \geq 0$ everywhere and $\frac{\partial^2 H}{\partial x_i \partial x_j} \leq 0$ almost everywhere. The function $h : R \times X \rightarrow \mathbb{R}$ defined by $h(S) = H[\xi(S)]$ is monotone and submodular.*

We turn to state the main result of this section. This result is later utilized to establish our hardness bound for monotone submodular maximization under a $(1, \ell)$ -partition matroid, and in particular under a binary partition matroid.

Theorem 5.8. *Let $\max\{f(S) : S \in \mathcal{I}\}$ be an instance invariant under a group of permutations \mathcal{G} , and exhibiting a symmetry gap of $\gamma = \overline{\text{OPT}}/\text{OPT}$. Let \mathcal{C} be the class of instances $\max\{\tilde{f}(S) : S \in \tilde{\mathcal{I}}\}$, where \tilde{f} is monotone and submodular, and $\tilde{\mathcal{I}}$ is a family of independent sets of a (k, ℓ) -partition matroid $\tilde{\mathcal{M}}$. For every $\epsilon > 0$, any $(1 + \epsilon)\gamma$ approximation algorithm for the class \mathcal{C} requires an exponential number of calls to \tilde{f} .*

Proof. Let $\epsilon > 0$. In what follows, we build two monotone submodular functions f and g over a r -blowup of \mathcal{M} , such that the optima of \tilde{f} and \tilde{g} under $\tilde{\mathcal{M}}$ differ by a factor of $(1+\epsilon)\gamma$. In addition, in order to differentiate between them (i.e., to find Q such that $\tilde{f}(Q) \neq \tilde{g}(Q)$), any algorithm has to make an exponential number of queries. Let $\max\{f(S) : S \in \mathcal{I}\}$ be the given instance. We apply Lemma 5.4 to get $\hat{F}, \hat{G} : [0, 1]^X \rightarrow \mathbb{R}$. Moreover, we choose a large natural number r and a random set of permutations $\{\sigma^{(i)}\}_{i=1}^r \subseteq \mathcal{G}$. The role of these permutations is to shuffle the labels of the copies of X , so that any algorithm does not “know” where the i -th copy of each element lies. Recall that Definition 5.6 presents a natural transformation from a subset $S \subseteq R \times X$ to a vector $\xi(S) \in [0, 1]^X$ using the chosen set of random permutations. We compose this transformation with \hat{F} and \hat{G} to obtain the functions $\tilde{f}, \tilde{g} : 2^{R \times X} \rightarrow \mathbb{R}$ defined as

$$\tilde{f}(S) = \hat{F}(\xi(S)), \tilde{g}(S) = \hat{G}(\xi(S))$$

Note that $\frac{\partial \hat{F}}{\partial x_i} \geq 0$ and $\frac{\partial^2 \hat{F}}{\partial x_i \partial x_j} \leq 0$, and therefore, we know that \tilde{f} is monotone and submodular by Lemma 5.7.

The same arguments hold also for \tilde{g} . Let $\tilde{\mathcal{I}}$ be the family of independent sets of the blowup matroid $\tilde{\mathcal{M}}$. Since \tilde{f} and \tilde{g} are both monotone and submodular, the two instances $\max\{\tilde{f}(S) : S \in \tilde{\mathcal{I}}\}$ and $\max\{\tilde{g}(S) : S \in \tilde{\mathcal{I}}\}$ belong to the class \mathcal{C} . Let $\text{OPT}_{\tilde{f}}$ and $\text{OPT}_{\tilde{g}}$ be their respective optimal values.

We now bound the ratio between the above-mentioned optimal values. One can validate that if $S \in \tilde{\mathcal{I}}$ then $\sum_{a_j \in P_t} \xi_j(S) \leq k$ for every partition subset P_t , and in turn, $\xi(S) \in \mathcal{P}(\mathcal{M})$. The former argument follows from the fact that $S \in \mathcal{I} \Leftrightarrow \sigma^{(i)}(S) \in \mathcal{I}$, for any permutation $\sigma^{(i)}$ under consideration. Thus, we obtain that

$$\text{OPT}_{\tilde{g}} = \max\{\tilde{g}(S) : S \in \tilde{\mathcal{I}}\} = \max\{\hat{G}(\xi(S)) : S \in \tilde{\mathcal{I}}\} \leq \max\{\hat{G}(x) : x \in \mathcal{P}(\mathcal{M})\} \leq (1+\epsilon)\overline{\text{OPT}},$$

where the last inequality holds by Lemma 5.4. On the other hand, let x^* be the optimal solution of $\max\{\hat{F}(x) : x \in \mathcal{P}(\mathcal{M})\}$. For a large enough r , we can approximate x^* with a rational vector $y^* \in \mathcal{P}(\mathcal{M})$ whose components have a common denominator r and y^* is arbitrarily close to x^* . The following lemma establishes that there is an independent set S in our blowup matroid $\tilde{\mathcal{M}}$ that corresponds to y^* .

Lemma 5.9. *There exists $S \in \tilde{\mathcal{I}}$ such that $\xi(S) = y^*$.*

Proof. Notice that the set of permutations $\{\sigma^{(i)}\}_{i=1}^r$ only renames the labels of the elements in each layer. Since our underlying instance is invariant under each $\sigma^{(i)}$, we can assume without loss of generality that it is the identity permutation.

We show an incremental procedure that builds a set S such that $\xi(S) = y^*$. The procedure considers each original partition subset $P_t = \{b_1, \dots, b_\ell\} \subseteq X$ and adds elements to S as follows. Let $q_1/r, \dots, q_\ell/r$ be the components of y^* corresponding to the elements of P_t . Add the elements

$$\left(1, b_1\right), \dots, \left(q_1, b_1\right), \left(1 + q_1, b_2\right), \dots, \left(q_1 + q_2, b_2\right), \dots, \left(1 + \sum_{j=1}^{\ell-1} q_j, b_\ell\right), \dots, \left(\sum_{j=1}^{\ell} q_j, b_\ell\right)$$

to S , where all the sums are taken modulo r .

Clearly, each $0 \leq q_j \leq r$ since $y^* \in [0, 1]^X$. This implies that the procedure only adds distinct elements to S , for each $b_j \in P_t$. Consequently, the procedure yields $S \subseteq R \times X$ such that $\xi(S) = y^*$. We now demonstrate that $S \in \tilde{\mathcal{I}}$. For this purpose, we need to verify that the procedure never selects more than k elements of each partition subset $P_{i,t}$. This follows because our procedure cycles through the layers corresponding to each original partition subset P_t , and the fact that $\sum_j q_j \leq kr$ since $y^* \in \mathcal{P}(\mathcal{M})$. ■

Recall that \hat{F} is a continuous function, and hence, $\tilde{f}(S) = \hat{F}(\xi(S)) = \hat{F}(y^*)$ approaches $\hat{F}(x^*)$ as r tends to infinity. In other words,

$$\text{OPT}_{\tilde{f}} = \max\{\tilde{f}(S) : S \in \tilde{\mathcal{I}}\} \rightarrow \max\{\hat{F}(x) : x \in \mathcal{P}(\mathcal{M})\}$$

as r goes to infinity. By Lemma 5.4, we know that $\max\{\hat{F}(x) : x \in \mathcal{P}(\mathcal{M})\} \geq \text{OPT}$. Consequently, we get that $\text{OPT}_{\tilde{g}}/\text{OPT}_{\tilde{f}}$ can be made arbitrarily close to a number which is at most $(1 + \epsilon)\gamma$. This shows that \tilde{f} and \tilde{g} exhibit a gap of $(1 + \epsilon)\gamma$ when constrained under $\tilde{\mathcal{M}}$.

To complete the proof, we need to show that any algorithm trying to differentiate between $\max\{\tilde{f}(S) : S \in \tilde{\mathcal{I}}\}$ and $\max\{\tilde{g}(S) : S \in \tilde{\mathcal{I}}\}$ must make an exponential number of oracle queries. This part follows the Chernoff's bound analysis of [27]. For sake of completeness, we present it here. Recall that $\{\sigma^{(i)}\}_{i=1}^r$ are chosen at random. This induces a distribution over the above-mentioned instances. We prove that any deterministic algorithm, given two instances sampled from this distribution, and making a subexponential number of oracle queries returns the same solution with high probability. By Yao's principle, it then follows that any randomized algorithm returns the same solution for some specific instances with high probability. Since we know that these instances have a gap of $(1 + \epsilon)\gamma$, it follows that no subexponential algorithm can guarantee an approximation ratio better than $(1 + \epsilon)\gamma$ for the problem $\max\{\tilde{f}(S) : S \in \tilde{\mathcal{I}}\}$.

Let \mathcal{A} be such a deterministic algorithm, and let $Q \subseteq R \times X$ be a query made by \mathcal{A} to the oracle. Let Q_{ij} to be the random indicator variable of the event $\{(i, \sigma^{(i)}(j)) \in Q\}$. By Definition 5.6,

$$\xi_j(Q) = \frac{1}{r} \sum_{i=1}^r Q_{ij}.$$

The expected value of Q_{ij} is equal to

$$\mathbb{E}[Q_{ij}] = \Pr_{\sigma^{(i)} \in \mathcal{G}} [(i, \sigma^{(i)}(a_j)) \in Q]$$

Summing over all $i \in [r]$, we obtain that

$$\mathbb{E}[\xi_j(Q)] = \frac{1}{r} \sum_{i=1}^r \mathbb{E}[Q_{ij}] = \frac{1}{r} \sum_{i=1}^r \Pr_{\sigma^{(i)} \in \mathcal{G}} [(i, \sigma^{(i)}(a_j)) \in Q] = \frac{1}{r} \mathbb{E}_{\sigma \in \mathcal{G}} [|\{i \in R : (i, \sigma(a_j)) \in Q\}|],$$

where the last equality results by noticing that since all $\sigma^{(i)}$ are drawn from the uniform distribution over \mathcal{G} , the expected value of $\mathbb{E}[Q_{ij}]$ is not effected if we draw them from a single permutation σ chosen uniformly. On the other hand, we can calculate the symmetrized version of $\xi_j(Q)$

$$\overline{\xi_j(Q)} = \mathbb{E}_{\sigma \in \mathcal{G}} [\xi_{\sigma(a_j)}(Q)] = \frac{1}{r} \mathbb{E}_{\sigma \in \mathcal{G}} [|\{i \in R : (i, \sigma^{(i)}(\sigma(a_j))) \in Q\}|] = \frac{1}{r} \mathbb{E}_{\sigma \in \mathcal{G}} [|\{i \in R : (i, \sigma(a_j)) \in Q\}|],$$

where the last equality holds since the uniform distribution on σ and the uniform distribution on $\sigma^{(i)} \circ \sigma$ are identical. We conclude that $\overline{\xi_j(Q)} = \mathbb{E}[\xi_j(Q)]$, for all j . Since the permutations $\{\sigma^{(i)}\}_{i=1}^r$ are chosen independently, we can apply Chernoff's bound (see, e.g., [1, Cor. A.1.7]) with respect to the random variables Q_{ij} , and obtain for all j that

$$\Pr \left[\left| \sum_{i=1}^r Q_{ij} - \sum_{i=1}^r \mathbb{E}[Q_{ij}] \right| > a \right] < 2e^{-2a^2/r}.$$

Choosing $a = r\sqrt{\delta/|X|}$, we obtain that $\Pr[|\xi_j(Q) - \overline{\xi_j(Q)}| > \sqrt{\delta/|X|}] < 2e^{-2r\delta/|X|}$. By applying union bound, we get

$$\Pr[|\|\xi(Q) - \overline{\xi(Q)}\|^2 > \delta] < 2|X|e^{-2r\delta/|X|}.$$

Notice that the right hand side probability diminishes exponentially with the size of r . Thus, with high probability, we have $\|\xi(Q) - \overline{\xi(Q)}\|^2 < \delta$. By Lemma 5.4, this implies that

$$\tilde{f}(Q) = \hat{F}[\xi(Q)] = \hat{G}[\xi(Q)] = \tilde{g}(Q)$$

with high probability. Since the probability that some query discovers a difference between \tilde{f} and \tilde{g} is exponentially small, we conclude that with high probability \mathcal{A} cannot differentiate between them and returns the same solution. ■

We now describe an instance of monotone submodular maximization under a $(1, \ell)$ -partition matroid that has a symmetry gap of $1 - (1 - 1/\ell)^\ell$. Consequently, we get an inapproximability result of the same ratio by Theorem 5.8. For the binary partition matroid case, this gives a $3/4$ -inapproximability result.

Theorem 5.10. *Any algorithm with an approximation ratio better than $1 - (1 - 1/\ell)^\ell$ to monotone submodular maximization under a $(1, \ell)$ -partition matroid must make an exponential number of oracle queries.*

Proof. By theorem 5.8, it suffices to show a specific instance that exhibits a symmetry gap of $1 - (1 - 1/\ell)^\ell$. Let $X = \{a_1, \dots, a_\ell\}$ be a ground set of elements, and let \mathcal{M} be a $(1, \ell)$ -partition matroid with one partition subset $P_1 = X$, namely, a uniform matroid. Moreover, let $f(S) = \min\{1, |S|\}$. Clearly, f is a monotone submodular function.

Now, let \mathcal{G} be the group of all permutations on X , and let F be the multilinear extension of f . Notice that $\text{OPT} = \max\{F(x) : x \in \mathcal{P}(\mathcal{M})\} = 1$ is obtained when x is any base of \mathcal{M} . One can easily validate that $\overline{\text{OPT}} = \max\{F(\bar{x}) : \bar{x} \in \mathcal{P}(\mathcal{M})\} = 1 - (1 - 1/\ell)^\ell$. This value is obtained if $\bar{x} = (1/\ell, \dots, 1/\ell)$. In this case, there is a probability of $(1 - 1/\ell)^\ell$ that $S = \emptyset$, and thus, $f(S) = 0$. ■

Corollary 5.11. *Any algorithm guaranteeing an approximation ratio better than $3/4$ to monotone submodular maximization under a binary partition matroid must make an exponential number of oracle queries.*

6 Concluding Remarks

In this paper, we introduced the submodular Max-SAT problem. We developed a randomized combinatorial linear-time $2/3$ -approximation algorithm that is applicable even for the online variant of the problem. We also established hardness results for both the online and offline variants of the problem: for the online setting, the hardness result proves that our algorithm is best possible; for the offline setting, the hardness result establishes a computational separation between the classical Max-SAT and the submodular Max-SAT.

A natural open question is to close the approximation gap for the offline version. In particular, this gap is between $2/3$ and $3/4$. We have recently learned that independently of our work, Feldman, Naor and Schwartz [12] considered the same problem and attained an improved approximation ratio. However, to the best of our knowledge, their algorithm is not fast and cannot be applied in an online setting.

References

- [1] N. Alon and J. H. Spencer. *The Probabilistic Method*. John Wiley & Sons, New York, 3rd edition, 2008.
- [2] T. Asano. Approximation algorithms for max sat: Yannakakis vs. goemans-williamson. In *Proceedings 5th Israel Symposium on Theory of Computing and Systems*, pages 24–37, 1997.
- [3] T. Asano. An improved analysis of goemans and williamson’s lp-relaxation for max sat. *Theor. Comput. Sci.*, 354(3):339–353, 2006.
- [4] T. Asano and D. P. Williamson. Improved approximation algorithms for max sat. *J. Algorithms*, 42(1):173–202, 2002.

- [5] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SICOMP*, 2010.
- [6] C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *Proceedings 51th Annual IEEE Symposium on Foundations of Computer Science*, 2010.
- [7] J. Chen, D. K. Friesen, and H. Zheng. Tight bound on johnson’s algorithm for maximum satisfiability. *J. Comput. Syst. Sci.*, 58(3):622–640, 1999.
- [8] D. Coppersmith, D. Gamarnik, M. T. Hajiaghayi, and G. B. Sorkin. Random max sat, random max cut, and their phase transitions. *Random Struct. Algorithms*, 24(4):502–545, 2004.
- [9] S. Dobzinski and M. Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *Proceedings 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1064–1073, 2006.
- [10] L. Engebretsen. Simplified tight analysis of johnson’s algorithm. *Inf. Process. Lett.*, 92(4):207–210, 2004.
- [11] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.
- [12] M. Feldman, J. Naor, and R. Schwartz. 2011. Personal Communication.
- [13] M. X. Goemans and D. P. Williamson. .878-approximation algorithms for max cut and max 2sat. In *Proceedings 26th Annual ACM Symposium on Theory of Computing*, pages 422–431, 1994.
- [14] M. X. Goemans and D. P. Williamson. New $3/4$ -approximation algorithms for the maximum satisfiability problem. *SIAM J. Discrete Math.*, 7(4):656–666, 1994.
- [15] P. R. Goundan and A. S. Schulz. Revisiting the greedy approach to submodular set function maximization. 2007. Manuscript.
- [16] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.
- [17] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974.
- [18] H. J. Karloff and U. Zwick. A $7/8$ -approximation algorithm for max 3sat? In *Proceedings 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 406–415, 1997.
- [19] A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 545–554, 2009.
- [20] J. Lee, M. Sviridenko, and J. Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. In *Proceedings 12th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 244–257, 2009.
- [21] V. S. Mirrokni, M. Schapira, and J. Vondrák. Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions. In *Proceedings 9th ACM Conference on Electronic Commerce*, pages 70–77, 2008.

- [22] D. Moshkovitz and R. Raz. Two-query pcp with subconstant error. *J. ACM*, 57(5), 2010.
- [23] G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Math. Operations Research*, 3(3):177–188, 1978.
- [24] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions I. *Mathematical Programming*, 14:265–294, 1978.
- [25] M. Poloczek and G. Schnitger. Randomized variants of johnson’s algorithm for max sat. In *Proceedings 22th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 656–663, 2011.
- [26] M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32(1):41–43, 2004.
- [27] J. Vondrák. Symmetry and approximability of submodular maximization problems. In *Proceedings 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 651–670, 2009.
- [28] L. A. Wolsey. Maximising real-valued submodular functions: Primal and dual heuristics for location problems. *Math. Operations Research*, 7(3):410–425, 1982.
- [29] M. Yannakakis. On the approximation of maximum satisfiability. *J. Algorithms*, 17(3):475–502, 1994.
- [30] U. Zwick. Outward rotations: A tool for rounding solutions of semidefinite programming relaxations, with applications to max cut and other problems. In *Proceedings 31st Annual ACM Symposium on Theory of Computing*, pages 679–687, 1999.