Partial Function Extension with Applications to Learning and Property Testing

Umang Bhaskar
Tata Institute of Fundamental Research, Mumbai, India
umang@tifr.res.in

Gunjan Kumar
Tata Institute of Fundamental Research, Mumbai, India
gunjan.kumar@tifr.res.in

Abstract

Partial function extension is a basic problem that underpins multiple research topics in optimization, including learning, property testing, and game theory. Here, we are given a partial function consisting of n points from a domain and a function value at each point. Our objective is to determine if this partial function can be extended to a function defined on the domain, that additionally satisfies a given property, such as linearity. We formally study partial function extension to fundamental properties in combinatorial optimization – subadditivity, XOS, and matroid independence. A priori, it is not clear if partial function extension for these properties even lies in NP (or coNP).

Our contributions are twofold. Firstly, for the properties studied, we give bounds on the complexity of partial function extension. For subadditivity and XOS, we give tight bounds on approximation guarantees as well. Secondly, we develop new connections between partial function extension and learning and property testing, and use these to give new results for these problems. In particular, for subadditive functions, we give improved lower bounds on learning, as well as the first subexponential-query tester.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Partial function extension, subadditivity, matroid rank, approximation algorithms, learning, property testing

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2020.46

Funding This research was supported by the Department of Atomic Energy, Government of India, under project no. RTI4001.
Umang Bhaskar: UB acknowledges support from a Ramanujan Fellowship (SERB – SB/S2/RJN-055/2015) and an Early Career Research Award (SERB – ECR/2018/002766) from the Government of India.
Gunjan Kumar: GK acknowledges support from a TCS fellowship.

1 Introduction

A *partial function* consists of a set $\mathcal{D}$ of points from a domain, and a real value at each of the points. Given a property P, the *partial function extension* problem is to determine if there exists a *total* function f (f is defined on the entire domain) that *extends* the partial function (f equals the given value at each point in $\mathcal{D}$) and satisfies P. E.g., property P could be linearity, and we are required to determine if there exists a linear function that extends the given partial function. In this paper, we study partial function extension when $\mathcal{D}$ is finite, to fundamental properties in combinatorial optimization.

The problem of partial function extension underpins research and techniques in a number of different areas. In *learning theory*, for example, the goal is to understand if functions with a given property can be learned by random samples. That is, does there exist an efficient algorithm that for any target function with the given property, takes as input the function values at a set of sampled points, and returns a function that is "close" to the target function? In learning theory the partial function extension problem is also known as the consistency problem. Pitt and Valiant [20] formally showed that the hardness of partial function extension for a class of functions can be used to show lower bounds on proper learning for this class, i.e., when the function returned must also belong to the same class. We use this connection in our paper as well. Partial function extension can also be used to give lower bounds on the learnability of various function classes beyond proper learning, and has been used thus in previous papers, e.g., [2]. Our lower bound for subadditive functions gives another example of this connection.

Partial function extension is used in *property testing* as well, where a function is given by an oracle, and the problem is to determine with high probability by querying the oracle whether the function satisfies a required property, or is far from it (e.g., [9, 22]). The focus in property testing is on algorithms with optimal query-complexity. A typical testing algorithm with one-sided error queries the values at points chosen from some distribution, and rejects iff the partial function given by the queried points and the values at these points cannot be extended to a function with the required property. Clearly partial function extension is useful both in design and analysis of property testing algorithms.

Besides these, partial function extension is studied in many other applications as well. For example, Topkis studies the problem of extending a partial function on a sublattice to a submodular function on the lattice, and applies it to obtaining conditions under which optimal solutions to an optimization problem are a monotone function of a parameter [24]. Extending partial functions to convex functions is widely studied in convex analysis [19, 27].

Partial function extension is thus a fundamental problem that finds many diverse applications. We focus on the complexity of deciding if a partial function can be extended to functions widely studied in combinatorial optimization – subadditive, XOS, and independence functions for matroids[1] – defined on $2^{[m]}$, the family of subsets of $\{1, \ldots, m\}$. Subadditive functions are important because they capture the notion of "complement-freeness", where the value of a set is no more than the sum of the values of constituent subsets. Subadditive functions are used, e.g., in game theory to model valuation functions of agents. XOS functions are a subclass of subadditive functions that have a natural interpretation – an XOS function is characterized by a number of linear functions, and the value at a point is the maximum over these linear functions. The support of an XOS function is the number of these linear functions over which the maximum is taken. Matroids generalize the notion of linear independence of vectors and acyclicity in graphs, and are thus basic and widely used combinatorial objects.

Partial Function Extension

Formally, a *partial function* is a set $H = \{(T_1, f_1), (T_2, f_2), \ldots, (T_n, f_n)\}$, with $T_i \in \{0,1\}^m$, and $f_i \in \mathbb{R}$ the observed function value at T_i. Additionally, we are given a property P. The *P-Extension* problem is to determine if there exists a total function f defined on the domain $\{0,1\}^m$ that satisfies property P and extends the given partial function H, i.e., $f(T_i) = f_i$ for all $i \in \{1, \ldots, n\}$. We also consider the *Approximate P-Extension* problem, where we want to determine the minimum multiplicative error for a given partial function to extend to a function that satisfies the given property. That is, in Approximate P-Extension, we want to approximate the minimum $\alpha \geq 1$ such that a function f satisfies property P and additionally, $f_i \leq f(T_i) \leq \alpha f_i$ for all $i \in \{1, \ldots, n\}$.

[1] These function classes are formally defined in the appropriate sections later.

Note that our input is H. An algorithm is efficient if it runs in time polynomial in the size of H, which may be exponential in the dimension m.

Proper Learning and Property Testing

The PMAC (Probably Mostly Approximate Correct) model seeks to determine for a family $\mathcal{F}$ of functions if it is possible to efficiently obtain a function f "close to" a target function $f^* \in \mathcal{F}$, given samples from some distribution over $2^{[m]}$ and the value of f^* at the sampled points. The learning is *proper* if the output function f is also in $\mathcal{F}$. Formally, let $\mathcal{F} \subseteq \{f : 2^{[m]} \to \mathbb{R}\}$ be a family of set functions (e.g., subadditive functions).

▶ **Definition 1** ([4]). *An algorithm $\mathcal{A}$ properly PMAC-learns a family of functions $\mathcal{F}$ with approximation factor α, if for* any *distribution μ (on $2^{[m]}$) and* any *target function $f^* \in \mathcal{F}$, and for* any *sufficiently small $\epsilon, \delta > 0$:*

- *$\mathcal{A}$ takes the sequence $\{(S_i, f^*(S_i))\}_{1\le i\le l}$ as input where l is $\text{poly}(m, 1/\delta, 1/\epsilon)$ and the sequence $\{S_i\}_{1\le i\le l}$ is drawn i.i.d. from the distribution μ*
- *$\mathcal{A}$ runs in $poly(m, 1/\delta, 1/\epsilon)$ time*
- *$\mathcal{A}$ returns a function $f : 2^{[m]} \to \mathbb{R} \in \mathcal{F}$ such that*

$$Pr_{S_1,\ldots,S_l\sim\mu}\left[Pr_{S\sim\mu}[f^*(S) \le f(S) \le \alpha f^*(S)] \ge 1-\epsilon\right] \ge 1-\delta$$

That is, with at least $1-\delta$ probability (over examples drawn from μ), the value of the returned function f should be within an α factor of the target function f^* for at least $1-\epsilon$ fraction of the probability mass according to μ.

If a family of functions is PMAC-learnable with $\alpha = 1$, then the family is said to be PAC-learnable. For Boolean functions, clearly PAC-learning and PMAC-learning are equivalent.

The following lemma makes an explicit connection between PMAC-learning and extending partial functions. The lemma has been implicitly used earlier to obtain lower bounds on learning submodular functions [4]. We also use this lemma to improve the previous bound on learning subadditive functions.

▶ **Lemma 2.** *Suppose there exists a family $\mathcal{D} = \{T_1, \ldots, T_n\}$ of subsets of $[m]$ such that n is superpolynomial in m, and the partial function $H = \{(T_1, f_1), \ldots, (T_n, f_n)\}$ is extensible to a function in $\mathcal{F}$ for any $f_i \in [1, r]$ (where $r \ge 1$), $i \in [n]$. Then the family of functions $\mathcal{F}$ cannot be learned by any factor $< r$.*

The lower bound given by the above lemma is information-theoretic, i.e., holds even if the algorithm knows the distribution μ, is allowed unbounded computation and chooses samples adaptively.

For a class of functions $\mathcal{F}$, a function $f : 2^{[m]} \to \mathbb{R}_+$ is ϵ-far from $\mathcal{F}$ if for any function $g \in \mathcal{F}$, we have $|\{S \subseteq [m] : f(S) \ne g(S)\}| \ge \epsilon 2^m$, i.e., g and f differ on at least $\epsilon 2^m$ points. A (one-sided) tester for $\mathcal{F}$ is a randomized algorithm that takes distance parameter ϵ and oracle access to a function $f : 2^{[m]} \to \mathbb{R}_+$ as inputs, and accepts if $f \in \mathcal{F}$, and rejects with constant probability if f is ϵ-far from $\mathcal{F}$.

We note a basic difference between partial function extension on the one hand, and property testing and learning on the other. In partial function extension, there is no target function f^*; we want to know if *any* total function that extends the given partial function has the required property. In property testing and learning, there is a target function f^* which we access via an oracle (in property testing) or via samples from a distribution (in learning).

We will frequently use reductions from MONOTONE-NAE-3SAT. Here, we are given a 3-SAT formula $\phi = C_1 \wedge \cdots \wedge C_{m'}$ with no negations of the variables. The problem is to determine if there exists an assignment (called *satisfying assignment*) such that at least one literal is true and at least one literal is false in each clause. The problem MONOTONE-NAE-3SAT is NP-hard [21]. We will assume that every clause contains three distinct variables.

For notation, $\mathcal{D} := \{T_i\}_{i\in[n]}$ is the set of points in the given partial function H. These are called *defined* points, and $\mathcal{U} := 2^{[m]} \setminus \mathcal{D}$ are *undefined* points. Points on the hypercube $\{0,1\}^m$ are naturally subsets of $[m]$, and for $S \subseteq [m]$, $\chi(S) \in \{0,1\}^m$ is its characteristic vector. We frequently use this correspondence.

Our Contribution

We give bounds on the complexity of partial function extension. For subadditivity and XOS, we give tight bounds on approximation guarantees as well. We then use these results to give bounds on the complexity of learning and testing all three function classes. Our lower bounds are for very simple functions in each class – XOS functions with just two linear functions in their support, and for any class of matroids that include graphic matroids.

XOS functions. We first show that in general, Approximate XOS Extension (and hence XOS Extension) can be determined in polynomial time. However, if we restrict the number of allowed linear functions for an XOS function, the extension problem becomes NP-hard, even for the case of two linear functions.

▶ **Theorem 3.** *Approximate XOS Extension is in P. However, for any $k \geq 2$, it is NP-hard to determine if there is an XOS function with k linear functions in its support that extends a given partial function.*

It is known that $\tilde{\Omega}(\sqrt{m})$ is a lower bound for PMAC-learning of XOS functions [2]. However, this lower bound requires superpolynomial support size for the XOS functions. Our hardness result for XOS extension allows us to show a lower bound for proper PAC learning of the much simpler class of XOS functions with just two linear functions in the support assuming $RP \neq NP$.[2]

▶ **Theorem 4.** *For any $k \geq 2$, the class of XOS functions with support size k cannot be properly PAC-learned unless $RP = NP$.*

Subadditive functions. We first show tight results for the complexity of Subadditive Extension.

▶ **Theorem 5.** *Subadditive Extension is coNP-complete. There is an $O(\log m)$ approximation algorithm for Approximate Subadditive Extension, and if $P \neq NP$, this is optimal.*

The lower bounds in the theorem depend upon characterizations of partial functions that can be extended to subadditive functions. The upper bound uses the fact that (a) for XOS functions, Approximate XOS Extension (and hence XOS Extension) can be solved in polynomial time, and (b) any subadditive function can be approximated by an XOS function by an $O(\log m)$ factor [7, 11].

[2] RP (Randomized Polynomial) is the class of problems for which a randomized algorithm runs in polynomial time, always answers correctly if the input is a "no" instance, and answers correctly with probability at least 1/2 if the input is "yes" instance.

The characterization for subadditive functions can be used to give the following results for learning. Our lower bound for learning improves upon a previous lower bound of $\Omega(\sqrt{m}/\log m)$ [2].

▶ **Theorem 6.** *Subadditive functions cannot be PMAC-learned by a $o(\sqrt{m})$ factor.*

Finally, we use the characterization for subadditive functions to give the first nontrivial tester for general (nonmonotone) subadditive functions.

▶ **Theorem 7.** *Given $\epsilon > 0$, there is a tester for general subadditive functions that makes $2^{O(\sqrt{m \log(1/\epsilon)} \log m)}$ queries.*

As a point of comparison, the square tester which is the first (and so far, the only) known tester for general submodular functions has query complexity $2^{O(\sqrt{m} \log m \log(1/\epsilon))}$ [22].

Independence functions for matroids. Given a matroid $M = (E, \mathcal{I})$ with ground set E and independent sets $\mathcal{I}$, the independence function $f^{\mathcal{I}} : 2^E \to \{0,1\}$ is a binary function on subsets of E that returns 1 if the subset is independent, and 0 otherwise. The rank function returns the size of the largest independent set in the subset. The two functions are polynomially equivalent, i.e., the value of one for a subset can be obtained by a polynomial number of calls to the other.

The Matroid Extension problem is to determine if there exists a matroid $(E, \mathcal{I})$ such that the independence function $f^{\mathcal{I}}(T_i) = f_i$ for all $i \in [n]$. We first show that extending a partial function to a matroid independence function is NP-hard. In fact, this is true for any class of matroids containing graphic matroids, including linear and regular matroids.

▶ **Theorem 8.** *Matroid Extension is NP-hard for any class of matroids containing graphic matroids.*

We use this result to show that the independence function for graphic matroids cannot be PAC-learned, unless RP = NP.

▶ **Theorem 9.** *Unless RP = NP, graphic matroids cannot be PAC-learned.*

Earlier work gives a lower bound of $\tilde{\Omega}(n^{1/3})$ on PMAC-learning of general matroid rank functions [4]. Our work on the other hand gives lower bounds for proper PAC-learning of matroid independence functions, for the well-studied class of graphic matroids. Despite the polynomial equivalence of independence and rank functions, due to differences inherent in the distribution, there is no known reduction from learning matroid rank functions to learning matroid independence functions.[3]

It can thus be seen that both for XOS and matroid independence functions, while our results for learning are for the more restrictive PAC learning (as compared to bounds on PMAC learning), we are able to show lower bounds for significantly simpler classes in comparison to previous work – for XOS functions with just two linear functions in the support, and for graphic matroids.

[3] In particular, to use matroid independence to learn the rank of a set would require the matroid independence learning algorithm to learn correctly the independence of particular subsets of the set, which is not guaranteed with high probability.

Related Work

The complexity of partial function extension to Boolean functions is studied earlier with various restrictions on the class of Boolean functions, such as size of formulae and occurrence of variables [20, 8]. Pitt and Valiant formally show that lower bounds on partial function extension can be used to show lower bounds for proper PAC learning [20]. In previous work, we study the complexity of partial function extension for two subclasses of subadditive functions: coverage functions and submodular functions. For coverage functions, we show that the partial function extension problem is NP-complete, and give bounds for Approximate Extension, and lower bounds for learning coverage functions [5]. For submodular functions, we give a new certificate of nonextendibility, and apply it to obtain results on extending submodular functions on lattices, and lower bounds for testing submodular functions [6].

In property testing of functions, the objective is to determine with high probability and with a sublinear (in the size of the domain) number of queries if the function satisfies a required property, or is ϵ-far from it, with distance defined appropriately. Bounds on property testers are known for many function properties, including convexity and submodularity [18]. For testing submodularity on the hypercube (i.e, $2^{[m]}$), a "square tester" is the best known with query complexity $O((1/\epsilon)^{\sqrt{m}\log m})$ [22]. A lower bound of $1/\epsilon^{4.8}$ is known on the number of queries required by the square tester.

For the problem of PMAC-learning submodular functions over sets of m elements, Balcan et al. show an upper bound of $O(\sqrt{m})$ and a lower bound of $\Omega(m^{1/3}/\log m)$ [4]. The lower bound is actually applicable to matroid rank functions, a subclass of submodular functions. For PMAC-learning subadditive functions, upper bounds of $O(\sqrt{m}\log m)$ and lower bounds of $\Omega(\sqrt{m}/\log m)$ are known [2]. The lower bound is shown for learning XOS functions, a class that lies between submodular and subadditive functions. The lower bound example requires XOS functions have superpolynomial support size. Better upper bounds are shown for learning XOS functions with small support size. If each XOS function in the family has support size k, then this class can be learned within a $O(k^\epsilon)$ factor in time $m^{O(1/\epsilon)}$ for any $\epsilon > 0$.

2 XOS Functions

A function $f : 2^{[m]} \to \mathbb{R}_+$ is an XOS function if it can be expressed as the maximum of k linear functions for some $k \geq 1$, i.e., there exist vectors $w^i \in \mathbb{R}_+^m$ for $1 \leq i \leq k$ such that $f(S) = \max \left(w^i\right)^T \chi(S)$ for every $S \subseteq [m]$. The vectors $w^i \in \mathbb{R}_+^m$ for $1 \leq i \leq k$ are called the *support* of the function f. XOS functions are subclasses of subadditive functions and are known to be equivalent to *fractionally subadditive* functions [14]. XOS functions for which k is bounded by a polynomial in m are called *succinct* XOS functions. Balcan et al. show that for any $\xi > 0$, succinct XOS functions can be PMAC-learned with $\alpha = k^\xi$ by an algorithm with running time $m^{1/\xi}$ [2]. Thus for constant k, XOS functions can be learned within a constant approximation factor in polynomial time. They also show a lower bound of $\tilde{\Omega}(\sqrt{m})$ for general XOS functions. For any fixed k, we show lower bounds for proper PAC learning of XOS functions with support size k.

We first show bounds on the complexity of partial function extension.

▶ **Theorem 3.** *Approximate XOS Extension is in P. However, for any $k \geq 2$, it is NP-hard to determine if there is an XOS function with k linear functions in its support that extends a given partial function.*

Proof. The positive result follows from writing a linear program with the vectors $(w^i)_{i\leq k}$ with $w^i \in \mathbb{R}^m_+$ as variables. While in general k may be exponential, a partial function H is extensible iff the linear program is feasible for $k = n$. The proof is in the appendix.

For the lower bound, we give a reduction from MONOTONE-NAE-3SAT. Given a 3-SAT formula $\phi = C_1 \wedge \cdots \wedge C_{m'}$, let the variables of ϕ be $x_1, \ldots, x_{n'}$.

We show here the proof for $k = 2$, and generalise it for $k > 2$ in the appendix. Given ϕ, we construct the partial function H as follows. The ground set is $[n']$. Our partial function consists of the n' sets $\{1\}, \ldots, \{n'\}$ each with value 1. Further for all clauses $C_j = x_{j1} \vee x_{j2} \vee x_{j3}$ ($j \in [m']$), the partial function is defined on $\{j_1, j_2, j_3\}$ with value 2. Thus there are $m' + n'$ defined sets in the partial function H.

Suppose ϕ has a satisfying assignment. We define the two vectors w^1 and w^2 in the support of the XOS function as follows. For $i \in [n']$, let $w^1_i = x_i$ and $w^2_i = 1 - x_i$. Clearly for any $i \in [n']$, we have $\max\{w^1_i, w^2_i\} = 1$. Further for any set $\{j_1, j_2, j_3\} \subseteq [n']$ (corresponding to $C_j = x_{j_1} \vee x_{j_2} \vee x_{j_3}$), since at least one variable has value 1 and at least one has value 0 in C_j, we have $\max\{w^1_{j_1} + w^1_{j_2} + w^1_{j_3}, w^2_{j_1} + w^2_{j_2} + w^2_{j_3}\} = 2$.

Now assume there is an XOS function f with $k = 2$ that extends the partial function. Let w^1 and w^2 be the vectors of two linear functions in the support. Let for all $i \in [n']$, $x_i = w^1_i$. Then $\max\{w^1_i, w^2_i\} = 1$ for all $i \in [n']$ (since $f(\{i\}) = 1$). Therefore, for any set $\{j_1, j_2, j_3\}$, at least one of $w^1_{j_1}, w^1_{j_2}, w^1_{j_3}$ must be 1 (since otherwise $w^2_{j_1} + w^2_{j_2} + w^2_{j_3} = 3$ contradicting $f(\{j_1, j_2, j_3\}) = 2$) and similarly not all $w^1_{j_1}, w^1_{j_2}, w^1_{j_3}$ can be 1. Thus, $x_i = w^1_i$ is a satisfying assignment. ◀

Next we show lower bounds for proper learning of XOS functions with support size k for any fixed $k \geq 2$. In the following we write $g(i)$ for $g(\{i\})$ and $g(i, j)$ for $g(\{i, j\})$. For fixed $k \geq 2$, let $\mathcal{F}$ be the family of XOS functions with k supports.

▶ **Theorem 4.** *For any $k \geq 2$, the class of XOS functions with support size k cannot be properly PAC-learned unless RP = NP.*

Proof. Recall the reduction in Theorem 3. Given ϕ (an instance of MONOTONE-NAE-3SAT), let $\mathcal{D}$ be the set of the defined points in the instance of the partial function, and let f be the values at the defined points. Let $\epsilon = \frac{1}{2|\mathcal{D}|}$ (and hence $\epsilon < 1/|\mathcal{D}|$) and μ be a uniform distribution over $\{(S, f(S)) | S \in \mathcal{D}\}$. Now suppose a (randomized) algorithm A PAC-learns $\mathcal{F}$. We will show that in this case, we can determine efficiently if f is extensible to a function in $\mathcal{F}$ and by Theorem 3 determine if ϕ has a satisfying assignment.

We present the proof for $k = 2$. For larger k, as in the reduction in Theorem 3, we simply restrict attention to the ground set $[n']$ and the partial function defined on its subsets. Suppose the input to the algorithm A is given by the distribution μ and let it return a function g. We now have oracle access to the function g. If the partial function constructed is extensible, then $g(S)$ must be an XOS function with two linear functions in its support, and further $g(S) = f(S)$ for all $S \in \mathcal{D}$ (since $\epsilon < 1/|\mathcal{D}|$ and A must satisfy$Pr_{S\sim\mu}[g(S) = f(S)] \geq 1 - \epsilon$). If the partial function constructed is not extensible, then clearly the two conditions cannot be satisfied (and the learning algorithm A must fail). We are thus left with the problem of verifying these two conditions in polynomial time, given oracle access to g. If g satisfies the two conditions, we will construct another function g' that satisfies these two conditions.

Let v^1, v^2 be the two linear functions in the support of g. Note that since in $\mathcal{D}$ each $\{i\}(i \in [n'])$ has value 1, for each $i \in [n']$, we have $\max\{v^1_i, v^2_i\} = 1$. We define the sets X', Y', Z' as subsets of $[n']$ as follows. Let $Y' = \{i : v^1_i = v^2_i = 1\}$, $X' = \{i : v^1_i = 1\} \setminus Y'$, and $Z' = \{i : v^2_i = 1\} \setminus Y'$. We will now obtain sets X, Y, Z and claim that $Y' = Y$, and either $X' = X$, $Z' = Z$ or $X' = Z$ and $Z' = X$. We then define the vectors $w^1 = \chi(X \cup Y)$ and

$w^2 = \chi(Y \cup Z)$, and the XOS function $g'(S) = \max\{(w^1)^T \chi(S), (w^2)^T \chi(S)\}$. It can be verified that if $g(S) = f(S)$ for $S \in \mathcal{D}$ then $g'(S) = f(S)$ for $S \in \mathcal{D}$, and hence $g'(S)$ satisfies the above conditions.

To obtain X, Y, and Z, we find the values of g at all sets of size at most two. Define $Y = \{i \in [n'] | \forall j \in [n'], g(i,j) = g(i) + g(j)\}$. If $Y = [n']$, then $X = Z = \emptyset$. Else, pick any $k \notin Y$, and let $X = \{k\} \cup \{i \in [n'] | g(i,k) = g(i) + g(k)\} \setminus Y$. Finally, let $Z = [n'] \setminus (X \cup Y)$.

We claim that $Y' = Y$, and either $X' = X$, $Z' = Z$ or $X' = Z$ and $Z' = X$. To see this, note that (i) for all $i, j \in X' \cup Y'$, $g(i,j) = g(i) + g(j)$, (ii) for all $i, j \in Z' \cup Y'$, $g(i,j) = g(i) + g(j)$, and (iii) for all $i \in X'$, $j \in Z'$, $g(i,j) < g(i) + g(j)$. The claim follows from the construction of X, Y, Z above. ◀

3 Subadditive Functions

We now consider the problem of extending a given partial function H to monotone subadditive functions. A function $f : 2^{[m]} \to \mathbb{R}_+$ is subadditive if $f(A) + f(B) \geq f(A \cup B)$ for all sets A and B, and monotone if $f(A) \geq f(B)$ for all $A \supseteq B$. Subadditive functions capture the important case of complement-free functions, for which no two subsets of the ground set $[m]$ “complement” each other. This is a natural assumption in many applications, and hence these functions and various subclasses, including XOS functions, are widely used in game theory [2, 7, 16].

We give the following characterization for subadditive functions, also implicit in Lemma 3.3 of [1].

▶ **Lemma 10** ([1]). *Partial function H is extensible to a subadditive function iff $\sum_{i=1}^{r} f(T_i) \geq f(T_{r+1})$ for all $T_1, \ldots, T_r, T_{r+1} \in \mathcal{D}$ such that $\cup_{i=1}^{r} T_i \supseteq T_{r+1}$.*

We use the characterization to show that $\Theta(\log m)$ is a tight bound on the approximability of Approximate Subadditive Extension, unless $P = NP$ (and that the Extension problem is coNP-complete). For the lower bound, we give a reduction from Set-Cover. For the upper bound, we use earlier results which show that any subadditive function can be $O(\log m)$-approximated by an XOS function [7, 11]. Since Approximate XOS Extension can be efficiently solved, this gives us our upper bound.

▶ **Theorem 5.** *Subadditive Extension is coNP-complete. There is an $O(\log m)$ approximation algorithm for Approximate Subadditive Extension, and if $P \neq NP$, this is optimal.*

A lower bound on learning subadditive functions

Balcan et al. [3] proved bounds of $O(\sqrt{m} \log m)$ and $\Omega(\sqrt{m}/\log m)$ lower bound for learning subadditive functions. Using Lemmas 2, 10, and a known result for cover-free families [13, 12], we show an improved lower bound of $\Omega(\sqrt{m})$.

▶ **Theorem 6.** *Subadditive functions cannot be PMAC-learned by a $o(\sqrt{m})$ factor.*

A family of sets $\mathcal{F} \subseteq 2^{[m]}$ is called an *r-cover free* family [13, 12] if for all distinct sets $A_1, \ldots, A_r, A_{r+1} \in \mathcal{F}$ we have $A_{r+1} \not\subseteq \cup_{i=1}^{r} A_i$. Let $f_r(m)$ be the cardinality of the largest r-cover free family.

▶ **Proposition 11** ([12]). $f_r(m) = 2^{\Theta\left(\frac{m \log r}{r^2}\right)}$.

▶ **Lemma 12.** *If $\mathcal{D} = \{T_1, \ldots, T_n\}$ is an r-cover free family then the partial function $\{(T_1, f_1), \ldots, (T_n, f_n)\}$ is extensible to a subadditive function for any values of $f_i \in [1, r+1], i \in [n]$.*

Proof. Suppose the partial function is not extensible. Therefore, by Lemma 10, there exists sets $T_1, \dots, T_k, T_{k+1}$ for some $k \geq 1$ such that $T_{k+1} \subseteq \cup_{i=1}^{k} T_i$ and $f_{k+1} > \sum_{i=1}^{k} f_i$. Therefore, we have $r + 1 \geq f_{k+1} > k$ which is a contradiction as $\mathcal{D}$ is an r-cover free family. ◀

Proof of Theorem 6. We have $f_r(m) \geq 2^{\frac{cm \log r}{r^2}} = m^{\frac{cm}{r^2}(\frac{1}{2} - \frac{\log(\sqrt{m}/r)}{\log m})}$ for some constant c. For $r \leq m^{1/4}$, $f_r(m)$ is clearly superpolynomial in m. Also, for $r \geq m^{1/4}$, $\frac{1}{2} - \frac{\log(\sqrt{m}/r)}{\log m} \geq \frac{1}{4}$. Hence, $2^{\frac{cm \log r}{r^2}}$ is superpolynomial for $r = o(\sqrt{m})$. Therefore, for any such r, by Proposition 11 there exists an r-cover free family $\mathcal{D} = \{T_1, \dots, T_n\}$ such that n is superpolynomial. The theorem is directly implied by Lemmas 2 and 12. ◀

A subexponential tester for general subadditive functions

We now describe a property testing algorithm for general (nonmonotone) subadditive functions that makes $2^{O(\sqrt{m \log(1/\epsilon)} \log m)}$ queries; in this subsection, subadditive refers to nonmonotone subadditive functions.

▶ **Theorem 7.** *Given $\epsilon > 0$, there is a tester for general subadditive functions that makes $2^{O(\sqrt{m \log(1/\epsilon)} \log m)}$ queries.*

Let $\lambda = \sqrt{\ln(4/\epsilon)}$, and define $M_\lambda = \{S \subseteq [m] | m/2 - \lambda\sqrt{m} \leq |S| \leq m/2 + \lambda\sqrt{m}\}$. The tester repeats the following steps $1/\epsilon$ times:

- Randomly pick a set $T \in M_\lambda$ and query the sets $Q = \{S \in M_\lambda | S \subseteq T\}$.
- If there exist $T_1, \dots, T_r \in Q$ for some $r \geq 1$ such that $T = \cup_{i=1}^{r} T_i$ and $f(T) > \sum_{i=1}^{r} f(T_i)$ then reject.

The tester makes $|Q|/\epsilon$ queries, and $|Q| \leq \binom{m/2+\lambda\sqrt{m}}{2\lambda\sqrt{m}}$, and hence $|Q| = 2^{O(\sqrt{m \log(1/\epsilon)} \log m)}$, which is also a bound on the number of queries by the tester. Obviously if the function f is subadditive then the tester accepts. For the proof of the theorem, we show that if f is ϵ-far from subadditive functions then the above tester rejects with constant probability. This is then the required tester. To prove this, we first give a characterization for partial function extension similar to Lemma 10 for general subadditive functions.

▶ **Lemma 13.** *The partial function H is extensible to a subadditive function iff $\sum_{i=1}^{r} f(T_i) \geq f(\cup_{i=1}^{r} T_i)$ for all $T_1, \dots, T_r \in \mathcal{D}$ such that $\cup_{i=1}^{r} T_i \in \mathcal{D}$.*

A set $T \in M_\lambda$ is called bad if it causes the tester to reject. The set of bad sets $\mathcal{B}$ consists of $T \in M_\lambda$ such that there exists $T_1, \dots, T_r \in M_\lambda$ for some $r \geq 1$ such that $T = \cup_{i=1}^{r} T_i$ and $f(T) > \sum_{i=1}^{r} f(T_i)$.

We show that removing all sets not in M_λ, as well as the bad sets, gives us a partial function that can be extended to subadditive function. Since the function is ϵ-far and M_λ is large by our choice of λ, there must be many bad sets.

▶ **Lemma 14.** *The partial function $H = \{(S, f(S)) | S \in M_\lambda \quad \text{and} \quad S \notin \mathcal{B}\}$ is extensible to a subadditive function.*

Proof. Suppose the partial function is not extensible. Let $\mathcal{D} = \{S | S \in M_\lambda \quad \text{and} \quad S \notin \mathcal{B}\}$ be the defined sets in H. Then by Lemma 13, there will exist $T_1, \dots, T_r, T \in \mathcal{D}$ such that $T = \cup_{i=1}^{r} T_i$ and $\sum_{i=1}^{r} f(T_i) < f(T)$. This implies $T \in \mathcal{B}$ which is a contradiction. ◀

Proof of Theorem 7. Let $\mathcal{D} = \{S | S \in M_\lambda \quad \text{and} \quad S \notin \mathcal{B}\}$ and $\mathcal{U} = 2^{[m]} \setminus \mathcal{D}$. Since the partial function $\{(S, f(S)) | S \in \mathcal{D}\}$ is extensible, $|\mathcal{U}| \geq \epsilon 2^m$ (since f is ϵ-far). Note that $|\mathcal{U}| = |\mathcal{B}| + 2\sum_{i=1}^{m/2-\lambda\sqrt{m}} \binom{m}{i}$. Hence again using Chernoff bound and the value of λ, we have $|\mathcal{B}| \geq \epsilon 2^m/2$. Therefore in a single iteration our tester will pick a bad set with $\epsilon/2$ probability. Hence after $1/\epsilon$ iterations, the tester will pick a bad set with constant probability. ◀

4 Independence Functions for Matroids

Matroids are basic combinatorial structures that generalize the notion of linear independence of vectors. In both graph theory and linear algebra, many important structures are matroids. For example, the set of forests in a graph form a graphic matroid. The set of linearly independent rows of a matrix form a linear matroid. Consequently, the study of matroids is a field of its own, and matroids find various applications including in combinatorial optimization, machine learning, and coding theory (e.g., [4, 15]).

Formally, a matroid M is a pair $(E, \mathcal{I})$ where E is a finite ground set and $\mathcal{I}$ is a family of subsets of E that satisfies: (1) $\mathcal{I}$ is non-empty, (2) if $A \subseteq B$ and $B \in \mathcal{I}$ then $A \in \mathcal{I}$, and (3) if $A, B \in \mathcal{I}$, and $|A| < |B|$ then there exists $e \in B \setminus A$ such that $A \cup e \in \mathcal{I}$. Members of $\mathcal{I}$ are called independent sets. By Property (3), all maximal independent sets have the same size. $M = (E, \mathcal{I})$ is a graphic matroid if there is a graph G with edges E so that if $I \in \mathcal{I}$, then the set of edges I is acyclic. We use $M(G)$ to refer to the graphic matroid defined by graph G. Matroid M is a linear matroid if there exist vectors $(v_i)_{i \in E}$ such that if $I \in \mathcal{I}$ then the set of vectors $(v_i)_{i \in I}$ are linearly independent. Further background on matroids is in [25].

The independent sets of a matroid naturally induce a Boolean function $f^{\mathcal{I}} : 2^E \to \{0, 1\}$ called the *independence function* where $f^{\mathcal{I}}(S) = 1$ iff S is an independent set. We are given a partial function $H = \{(T_1, f_1), (T_2, f_2), \ldots, (T_n, f_n)\}$ where $T_i \subseteq E, f_i \in \{0, 1\}$ for all $i \in [n]$. The Matroid Extension problem is to determine if there exists a matroid $(E, \mathcal{I})$ such that $f^{\mathcal{I}}(T_i) = f_i$ for all $i \in [n]$. If the independence function of a matroid extends a partial function, we say that the partial function is extensible to a matroid. We also consider extension to restricted classes, including graphic and linear matroids.

We show the following sweeping negative result.

▶ **Theorem 8.** *Matroid Extension is NP-hard for any class of matroids containing graphic matroids.*

Important classes of matroids that contain graphic matroids are linear, binary, ternary and regular matroids [17]. By Theorem 8, the extension problem is NP-hard for all of these.

For the proof of Theorem 8, we first give some well-established properties of matroids. A set $S \subseteq E$ is called *dependent* if it is not an independent set. A minimal dependent set is called a *circuit.*

▶ **Theorem 15** ([26, 17])**.** *Let $M = (E, \mathcal{I})$ be a matroid. If $I \in \mathcal{I}$ and $I \cup e \notin \mathcal{I}$ then $I \cup e$ contains a unique circuit.*

▶ **Theorem 16** ([26, 17])**.** *Let $\mathcal{C}$ be the set of circuits of a matroid. If $C_1, C_2 \in \mathcal{C}, C_1 \neq C_2$ and $e \in C_1 \cap C_2$, then $C_1 \cup C_2 \setminus e$ contains a circuit.*

Proof of Theorem 8. The proof is by reduction from MONOTONE-NAE-3SAT. Suppose we are given an instance $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_{m'}$ of MONOTONE-1-IN-3SAT with n' variables and m' clauses. We assume that the variables are $x_1, x_2, \ldots, x_{n'}$. In our reduction, we show that if ϕ is satisfiable then a graphic matroid extends the partial function and if ϕ is unsatisfiable then no matroid extends the partial function. This would prove Theorem 8. Below we give the construction of the instance of Matroid Extension from ϕ.

The ground set for the Matroid Extension instance consists of $m = 4n'$ elements, with a_i, b_i, c_i, d_i associated with each variable x_i. We will denote the ground set by E. The partial function H is defined on $4n' + 2m' + 1$ sets, and is shown in Table 1. For each variable x_i, each of the 3 sets $\{a_i, b_i, d_i\}$, $\{a_i, b_i, c_i\}$, and $\{c_i, d_i\}$ are independent sets (and hence the

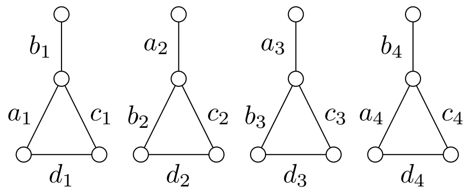

Figure 1 The graph G for the satisfying assignment $x_1 = x_4 = 1, x_2 = x_3 = 0$ of the instance $\phi = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee x_4)$.

Table 1 The partial function H for the instance $\phi = C_1 \wedge \cdots \wedge C_{m'}$ where $C_j = (x_{j1} \vee x_{j2} \vee x_{j3})$ $(j \in [m'])$.

T	f
$\{a_1, \ldots, a_{n'}, b_1, \ldots, b_{n'}, c_1, \ldots, c_{n'}\}$	1
$\{a_i, b_i, c_i, d_i\}$	0
$\{a_i, b_i, d_i\}$, $\{a_i, b_i, c_i\}$, $\{c_i, d_i\}$	1
$\{a_{j1}, c_{j1}, d_{j1}, a_{j2}, c_{j2}, d_{j2}, a_{j3}, c_{j3}, d_{j3}\}$	0
$\{b_{j1}, c_{j1}, d_{j1}, b_{j2}, c_{j2}, d_{j2}, b_{j3}, c_{j3}, d_{j3}\}$	0

function values are 1). The set $\{a_i, b_i, c_i, d_i\}$ is dependent (and hence has value 0). The set $\cup_{i \in [n']}\{a_i, b_i, c_i\}$ is independent. For the clause $C_j = (x_{j1} \vee x_{j2} \vee x_{j3})$, we include the dependent sets $\cup_{i \in [3]}\{a_{ji}, c_{ji}, d_{ji}\}$ and $\cup_{i \in [3]}\{b_{ji}, c_{ji}, d_{ji}\}$ in the partial function. Thus, the set $\mathcal{D}$ of defined sets has size $2m' + 4n' + 1$.

First assume that there exists a satisfying assignment of ϕ. We construct a graphic matroid $M(G)$ consistent with the partial function. We construct a graph G (see Figure 1) with n' connected components (one for each variable) as follows: If $x_i = 1$ then the ith component is a graph with 4 edges a_i, b_i, c_i, d_i such that a_i, c_i, d_i forms a simple cycle. Otherwise (if $x_i = 0$) the ith component is a graph with 4 edges a_i, b_i, c_i, d_i such that b_i, c_i, d_i forms a simple cycle. Now we show that the independence function of this matroid is consistent with the partial function. Since for each $i \in [n]$, the edges a_i, b_i, c_i, d_i contains a cycle so the sets $\{a_i, b_i, c_i, d_i\}$ are not independent. Also, since cycles are formed only by the edges $\{a_i, c_i, d_i\}$ or by $\{b_i, c_i, d_i\}$ (and not by edges $\{c_i, d_i\}$), the sets $\{a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n\}$, $\{a_i, b_i, d_i\}$, $\{a_i, b_i, c_i\}$ and $\{c_i, d_i\}$ are independent.

For all clauses $C_j = \{x_{j1} \vee x_{j2} \vee x_{j3}\}$, at least one variable is true and one variable is false, so the edge sets $\{a_{j1}, c_{j1}, d_{j1}, a_{j2}, c_{j2}, d_{j2}, a_{j3}, c_{j3}, d_{j3}\}$ and $\{b_{j1}, c_{j1}, d_{j1}, b_{j2}, c_{j2}, d_{j2}, b_{j3}, c_{j3}, d_{j3}\}$ contain a cycle, and hence are not independent.

Suppose now that H is extensible to a matroid $M = (E, \mathcal{I})$. We will show a satisfying assignment of ϕ. Let $\mathcal{C}$ be the set of circuits of this matroid M. Since $\{a_i, b_i, d_i\} \in \mathcal{I}$ and $\{a_i, b_i, c_i, d_i\} \notin \mathcal{I}$, by Theorem 15 $\{a_i, b_i, c_i, d_i\}$ contains a unique circuit. This means that exactly one of $\{a_i, c_i, d_i\}, \{b_i, c_i, d_i\}, \{a_i, b_i, c_i, d_i\}$ is a circuit for all $i \in [n']$. Consider an assignment in which x_i is set to 1 if $\{a_i, c_i, d_i\}$ is a circuit in $\mathcal{C}$ and 0 otherwise. Our claim is that this assignment is a satisfying assignment. Consider any clause $C_j = \{x_{j1} \vee x_{j2} \vee x_{j3}\} (j \in [m])$. Recall that we assumed that the variables appearing in any clause are distinct. Thus b_{j1}, b_{j2}, and b_{j3} are all distinct. First we show all x_{j1}, x_{j2}, x_{j3} cannot be 0. We will show this by contradiction, that if indeed all of x_{j1}, x_{j2}, x_{j3} are 0, then there is a circuit $\bar{C}$ that does not contain d_{j3} (and d_{j1}, d_{j2}). This contradicts that $\{a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n\} \in \mathcal{I}$.

Suppose all x_{j1}, x_{j2} and x_{j3} are 0. Since $\{a_{j1}, c_{j1}, d_{j1}, a_{j2}, c_{j2}, d_{j2}, a_{j3}, c_{j3}, d_{j3}\} \notin \mathcal{I}$, it contains a circuit C_1. If C_1 does not contain d_{j1}, d_{j2}, or d_{j3}, then let $\bar{C} = C_1$, and we are done. Otherwise suppose $d_{j1} \in C_1$, and note that $b_{j1} \notin C_1$.

Since $x_{j1} = 0$, Theorem 15 implies that exactly one of $\{a_{j1}, b_{j1}, c_{j1}, d_{j1}\}$ and $\{b_{j1}, c_{j1}, d_{j1}\}$ is a circuit. Let $\{a_{j1}, b_{j1}, c_{j1}, d_{j1}\}$ be a circuit (the proof works exactly the same way if $\{b_{j1}, c_{j1}, d_{j1}\}$ is a circuit). Since circuit C_1 does not contain b_{j1}, it is distinct from the circuit $\{a_{j1}, b_{j1}, c_{j1}, d_{j1}\}$. Then by Theorem 16, $C' = C_1 \cup \{a_{j1}, b_{j1}, c_{j1}, d_{j1}\} \setminus d_{j1}$ contains a circuit C_2. Note that C_2 is distinct from the circuit $\{a_{j2}, b_{j2}, c_{j2}, d_{j2}\}$ (C_2 does not contain b_{j2} and $b_{j1} \neq b_{j2}$). If $d_{j2} \in C_2$ then $C_2 \cup \{a_{j2}, b_{j2}, c_{j2}, d_{j2}\} \setminus d_{j2}$ contains a circuit C_3 (if $d_{j2} \notin C_2$ then let $C_3 = C_2$). Again C_3 does not contain b_{j3} and as before if $d_{j3} \in C_3$, then we can get a circuit $\bar{C}$ that does not contain d_{j3} (and d_{j1}, d_{j2}). This contradicts that $\{a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n\} \in \mathcal{I}$.

Thus at least one of x_{j1}, x_{j2}, x_{j3} must be 1. Now we show all of them cannot be 1. Suppose $x_{j1} = x_{j2} = x_{j3} = 1$. Therefore, $\{a_{j1}, c_{j1}, d_{j1}\}, \{a_{j2}, c_{j2}, d_{j2}\}$ and $\{a_{j3}, c_{j3}, d_{j3}\}$ are circuits. Our argument for this case is very similar to the previous case. We have $\{b_{j1}, c_{j1}, d_{j1}, b_{j2}, c_{j2}, d_{j2}, b_{j3}, c_{j3}, d_{j3}\} \notin \mathcal{I}$, and so it contains a circuit D_1. The circuit D_1 must contain some of d_{j1}, d_{j2}, d_{j3} otherwise $D_1 \in \mathcal{I}$ (by property (2) of matroids since $\{a_1, \ldots, a_n, b_1, \ldots, b_n,\ c_1, \ldots, c_n\} \in \mathcal{I}$). Suppose $d_{j1} \in D_1$. As noted before, $\{a_{j1}, c_{j1}, d_{j1}\}$ is a circuit. Since circuit D_1 does not contain a_{j1}, it is distinct from $\{a_{j1}, c_{j1}, d_{j1}\}$. Then by Theorem 16, $D' = D_1 \cup \{a_{j1}, c_{j1}, d_{j1}\} \setminus d_{j1}$ contains a circuit D_2. Note that D_2 is distinct from the circuit $\{a_{j2}, c_{j2}, d_{j2}\}$ (D_2 does not contain a_{j2} and $a_{j1} \neq a_{j2}$). If $d_{j2} \in D_2$ then $D_2 \cup \{a_{j2}, c_{j2}, d_{j2}\} \setminus d_{j2}$ contains a circuit D_3. Again D_3 does not contain a_{j3} and as before, if $d_{j3} \in D_3$, then we can get a circuit $\bar{D}$ that does not contain d_{j3} (and d_{j1}, d_{j2}). This contradicts that $\{a_1, \ldots, a_n, b_1, \ldots, b_n,\ c_1, \ldots, c_n\} \in \mathcal{I}$. ◀

We now show that unless RP=NP, PAC learning is not possible for the independence function of graphic matroids. We will use a reconstruction algorithm by Seymour for graphic matroids [23]. Here, the independence function for a matroid $M = (E, \mathcal{I})$ is given by an oracle that takes a set $S \subseteq [m]$ as input and returns 1 or 0 depending on whether S is an independent set in M or not. Seymour shows an algorithm that terminates in time poly($|E|$) and, if M is a graphic matroid, uses the oracle to find a graph G such that $M = M(G)$. We call this algorithm as the *graph recognizing* algorithm. We use Theorem 8 and the graph recognizing algorithm to show that graphic matroids cannot be PAC-learned.

▶ **Theorem 9.** *Unless RP = NP, graphic matroids cannot be PAC-learned.*

5 Conclusion

Our work is the first to study the complexity of partial function extension for fundamental classes of functions: subadditive, XOS, and matroid independence. For subadditive and XOS functions, the bounds we obtain are tight, and we utilise our results to obtain new and improved bounds for learning and property testing of these function classes. Besides these connections to learning and property testing, we consider the problem of partial function extension interesting in its own right, and with applications to many other areas, such as combinatorial optimization [24]. In previous work [5, 6] we study partial function extension for coverage and submodular functions. We believe the study of partial function extension to be a rich area, a focused study of which is likely to reveal many interesting properties of these function classes, as well as results on many related problems. A particularly interesting question left open by our work is the complexity of partial function extension for submodular functions. The question is alluded to in a number of papers, e.g., [4, 22], however the complexity of the problem remains unresolved.

References

1 Ashwinkumar Badanidiyuru, Shahar Dobzinski, Hu Fu, Robert Kleinberg, Noam Nisan, and Tim Roughgarden. Sketching valuation functions. In *SODA*, pages 1025–1035, 2012.

2 Maria-Florina Balcan, Florin Constantin, Satoru Iwata, and Lei Wang. Learning valuation functions. In *COLT 2012, June 25-27, 2012, Edinburgh, Scotland*, pages 4.1–4.24, 2012.

3 Maria Florina Balcan, Florin Constantin, Satoru Iwata, and Lei Wang. Learning valuation functions. In *Conference on Learning Theory*, pages 4–1, 2012.

4 Maria-Florina Balcan and Nicholas J. A. Harvey. Learning submodular functions. In *STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 793–802, 2011.

5 Umang Bhaskar and Gunjan Kumar. The complexity of partial function extension for coverage functions. In *APPROX/RANDOM Conference*, volume 145 of *LIPIcs*, pages 30:1–30:21, 2019.

6 Umang Bhaskar and Gunjan Kumar. A non-extendibility certificate for submodularity and applications. In *International Computing and Combinatorics Conference*, pages 603–614. Springer, 2020.

7 Kshipra Bhawalkar and Tim Roughgarden. Welfare guarantees for combinatorial auctions with item bidding. In *SODA*, pages 700–709, 2011.

8 Endre Boros, Toshihide Ibaraki, and Kazuhisa Makino. Error-free and best-fit extensions of partially defined Boolean functions. *Inf. Comput.*, 140(2):254–283, 1998.

9 Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *STOC*, pages 419–428, 2013.

10 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *STOC 2014*, pages 624–633. ACM, 2014.

11 Shahar Dobzinski. Two randomized mechanisms for combinatorial auctions. In *APPROX-RANDOM*, volume 4627 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2007.

12 Arkadii G D'yachkov, Il'ya Viktorovich Vorob'ev, NA Polyansky, and V Yu Shchukin. Bounds on the rate of disjunctive codes. *Problems of Information Transmission*, 50(1):27–56, 2014.

13 Paul Erdös, Peter Frankl, and Zoltán Füredi. Families of finite sets in which no set is covered by the union ofr others. *Israel Journal of Mathematics*, 51(1):79–89, 1985.

14 Uriel Feige. On maximizing welfare when utility functions are subadditive. *SIAM Journal on Computing*, 39(1):122–142, 2009.

15 Navin Kashyap, Emina Soljanin, and Pascal Vontobel. Applications of matroid theory and combinatorial optimization to information and coding theory, 2009.

16 Benny Lehmann, Daniel J. Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296, 2006.

17 James Oxley. What is a matroid. *Cubo Matemática Educacional*, 5(3):179–218, 2003.

18 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. On testing convexity and submodularity. *SIAM J. Comput.*, 32(5):1158–1184, 2003.

19 Hans JM Peters and Peter P Wakker. Convex functions on non-convex domains. *Economics letters*, 22(2-3):251–255, 1986.

20 Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *J. ACM*, 35(4):965–984, 1988.

21 Thomas J Schaefer. The complexity of satisfiability problems. In *STOC 1978*, pages 216–226. ACM, 1978.

22 C. Seshadhri and Jan Vondrák. Is submodularity testable? *Algorithmica*, 69(1):1–25, 2014.

23 Paul D. Seymour. Recognizing graphic matroids. *Combinatorica*, 1(1):75–78, 1981.

24 Donald M. Topkis. Minimizing a submodular function on a lattice. *Operations Research*, 26(2):305–321, 1978.

25 Dominic JA Welsh. *Matroid theory.* Courier Corporation, 2010.

26 Hassler Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57(3):509–533, 1935.

27 William K Wootters. Entanglement of formation of an arbitrary state of two qubits. *Physical Review Letters*, 80(10):2245, 1998.

A Appendix

Proof of Lemma 2

Consider the distribution μ that assigns probability mass uniformly to $\mathcal{D} = \{T_1, \dots, T_n\}$ and 0 elsewhere. We restrict the target function to the family $\mathcal{F}' = \{f \in \mathcal{F} | f(T_i) \in [1, r] \, \forall i \in [n]\} \subseteq \mathcal{F}$. Then for any algorithm that PMAC-learns $\mathcal{F}'$ with approximation factor $< r$, and for $S \sim \mu$, if f is the function learned by the algorithm and f^* is the target function, then $f^*(S) \le f(S) \le \alpha f^*(S)$ for $\alpha < r$. This is only possible if S is seen in the learning phase, i.e., the samples must be a constant fraction of n, and hence superpolynomial. ◀

A.1 XOS and Subadditive functions

Proof of Theorem 3. To show the upper bound, we prove that Approximate XOS extension is in P, and that if there is an extension then there exists an extension with support size at most n. Let the given partial function be $H = \{(T_1, f_1), \dots, (T_n, f_n)\}$. We claim that the optimal value of α for Approximate XOS Extension (say $\hat{\alpha}$) is equal to the optimal value of α in the following linear program (say α^*), with variables α and w_{ij} for all $1 \le i \le n$ and $1 \le j \le m$. Since the linear program can be solved in polynomial time, this claim implies that Approximate XOS Extension can be efficiently solved.

$$\min \alpha$$

$$f_i \le w_i^T \chi(T_i) \le \alpha f_i \quad \forall i \in [n]$$

$$w_i^T \chi(T_i) \ge w_j^T \chi(T_i) \quad \forall i, j \in [n]$$

$$w_i \in \mathbb{R}_+^m$$

$$\alpha \ge 1$$

To prove the claim, let the XOS function g corresponding to the optimal solution $\hat{\alpha}$ for Approximate XOS Extension be given by linear functions $v_1, \dots, v_k \in \mathbb{R}_+^m$ for some $k \ge 1$, and let the linear functions be indexed so that $g(T_i) = v_i^T \chi(T_i)$ for $i \in [n]$ (the same linear function can appear with multiple indices, i.e., $v_i = v_j$ for $i \neq j$). Then $f_i \le g(T_i) = v_i^T \chi(T_i) \le \hat{\alpha} f_i$ for all $i \in [n]$, and $v_i^T \chi(T_i) \ge v_j^T \chi(T_i)$ for all $i, j \in [n]$. It is clear that $\hat{\alpha}$, $\langle v_i \rangle_{i \in [n]}$ are feasible for the linear program, hence $\alpha^* \le \hat{\alpha}$. By definition, $\hat{\alpha} \le \alpha^*$, since the linear program produces an XOS function that has value within α^* factor at each T_i for all $i \in [n]$. Hence $\hat{\alpha} = \alpha^*$.

To complete the proof for the lower bound, we consider the case that $k > 2$. Let $k' = k - 2$. Then in the above reduction, we additionally include k' additional elements $n' + 1, \dots, n' + k'$ in the ground set. Each of these additional elements is included as a separate point in our partial function, with a large value $100(n' + k')$. Additionally, we include the set $[n' + k']$ in the partial function with value $100(n' + k')$. It is clear that these additional sets in the partial function necessitate k' additional linear functions in the support, each of which has a single coefficient (corresponding to one of the sets $\{n' + 1\}, \dots, \{n' + k'\}$) equal to $100(n' + k')$, and all other coefficients 0. The remainder of the proof follows as for the case $k = 2$. ◀

Proof of Lemma 10. For the first direction, if there exist $T_1, \dots, T_r, T_{r+1} \in \mathcal{D}$ such that $\cup_{i=1}^r T_i \supseteq T_{r+1}$ and $\sum_{i=1}^r f(T_i) < f(T_{r+1})$ then either $f(T_{r+1}) > f(\cup_{i=1}^r T_i)$ or $f(\cup_{i=1}^r T_i) > \sum_{i=1}^r f(T_i)$, and hence either monotonicity or subadditivity is violated. For the other direction, first assume that $\cup_{i=1}^n T_i = [m]$. Then the function $\hat{f}(S) = \{\min \sum_{T \in \mathcal{T}} f(T) | S \subseteq \cup_{T \in \mathcal{T}} T,\ \mathcal{T} \subseteq \mathcal{D}\}$ is an extension, monotone and subadditive. If $\cup_{i=1}^n T_i \subsetneq [m]$ then the function $\tilde{f}$ is a monotone subadditive extension, where $\tilde{f}(S)$ is defined as: $\tilde{f}(S) = \hat{f}(S)$ for all $S \subseteq \cup_{i=1}^n T_i$, and otherwise equal to $\hat{f}(S')$ where $S' = S \bigcap \cup_{i=1}^n T_i$. ◀

Proof of Theorem 5. Recall the Set-Cover problem. An instance of Set-Cover is a universe $[m]$, family of sets $V = \{S_1, \dots, S_n\}$ such that $S_i \subseteq [m]$ and an integer k. We need to determine if there exists a cover of universe $[m]$ of size at most k.

First we prove that the Extension problem is coNP-hard by reduction from Set-Cover. Construct a partial function that is defined on each set in V and $[m]$. The value at each set in V is 1 and at $[m]$ is $k+1$. If this partial function can be extended then every cover of $[m]$ must have size at least $k+1$. On the other hand, if partial function can not be extended then there must exist a cover of size at most k. Both of the above facts easily follow from Lemma 10.

For the lower bound of $\Omega(\log m)$ for Approximate Extension, as before, the partial function is defined on sets $V \cup [m]$, and the value at each set in V is 1, and at $[m]$ is m. Suppose we have an α-approximation algorithm for Approximate Extension, which for this instance returns value β. Note that $\epsilon^* \geq \beta/\alpha$ where ϵ^* is the optimal value of ϵ in the Approximate Extension problem.

Since the algorithm returns value β, so there exists an extension f such that $1 \leq f(S_i) \leq \beta$ for all $S_i \in V$ and $f([m]) \geq m$. Therefore, by Lemma 10, every cover of $[m]$ has size at least m/β. Now we claim that there must exist a cover with size at most $m\alpha/\beta$. Otherwise if all covers of $[m]$ has size at least $\gamma > m\alpha/\beta$ then it is easy to see that the partial function $\{(S_1, m/\gamma), \ldots, (S_n, m/\gamma), ([m], m)\}$ is extensible by Lemma 10. This implies $m/\gamma \geq \epsilon^* \geq \beta/\alpha$ which is a contradiction. This then gives an α-approximation algorithm for Set Cover, and since Set Cover cannot be approximated by a factor better than $(1-\epsilon)\log m$ [10], this is true of Approximate Extension also.

We now show the upper bound for Approximate Extension. Let $\mathcal{F}$ and $\mathcal{G}$ be two classes of functions. We say that $\mathcal{G}$ θ-approximates $\mathcal{F}$ if for all functions f in $\mathcal{F}$, there exists a function g in $\mathcal{G}$ such that $g(S) \leq f(S) \leq \theta g(S)$ for all $S \subseteq [m]$. We first prove the following lemma.

▶ **Lemma 17.** *Let $\mathcal{F}$ and $\mathcal{G}$ be two classes of functions so that $\mathcal{G}$ θ_1-approximates $\mathcal{F}$ and $\mathcal{F}$ θ_2-approximates $\mathcal{G}$. If there is a ρ-approximation algorithm for Approximate Extension for $\mathcal{F}$ then there is an $\rho\theta_1\theta_2$- approximation algorithm for Approximate Extension for $\mathcal{G}$.*

Proof. For a given instance of partial function extension, let $\alpha^*_{\mathcal{F}}$ and $\alpha^*_{\mathcal{G}}$ be the optimal values of α in the Approximate Extension problem for $\mathcal{F}$ and $\mathcal{G}$ respectively. Let A be the ρ-approximation algorithm for $\mathcal{F}$. A $\rho\theta_1\theta_2$-approximation algorithm for Approximate Extension for $\mathcal{G}$ is as follows: given any partial function H, return $\theta_1\alpha$ where α is the value returned by algorithm A on H. We have $\alpha^*_{\mathcal{F}} \geq \frac{\alpha}{\rho}$ as A is a ρ-approximation algorithm. Since $\mathcal{G}$ θ_1-approximates $\mathcal{F}$, we have $\alpha^*_{\mathcal{G}} \leq \theta_1\alpha^*_{\mathcal{F}}$. As $\alpha^*_{\mathcal{F}} \leq \alpha$ so we have $\alpha^*_{\mathcal{G}} \leq \theta_1\alpha$. Also $\mathcal{F}$ θ_2-approximates $\mathcal{G}$ so we have $\alpha^*_{\mathcal{F}} \leq \theta_2\alpha^*_{\mathcal{G}}$. Then $\alpha^*_{\mathcal{G}} \geq \frac{\alpha^*_{\mathcal{F}}}{\theta_2} \geq \frac{\alpha}{\rho\theta_2}$. Hence $\frac{\alpha}{\rho\theta_2} \leq \alpha^*_{\mathcal{G}} \leq \theta_1\alpha$. This proves our result. ◀

Recall that XOS functions are a subclass of subadditive functions. We will use the following result:

▶ **Theorem 18** ([7, 11]). *For any subadditive function f, there exists an XOS function g such $g(S) \leq f(S) \leq O(\log m)g(S)$ for all $S \subseteq [m]$.*

From the above results and Lemma 17, the upper bound for Theorem 5 follows, with $\theta_1 = 1, \theta_2 = O(\log m)$ and $\rho = 1$. ◀

Proof of Lemma 13. One direction is trivial. If there exist $T_1, \ldots, T_r, \cup_{i=1}^r T_i \in \mathcal{D}$ such that $\sum_{i=1}^r f(T_i) < f(\cup_{i=1}^r T_i)$ then partial function is not extensible to a general subadditive functions. Now assume this is not the case. Let $\mathcal{D}^c := \{S | S = \cup_{i=1}^r A_i \text{for some} A_1, \ldots, A_r \in \mathcal{D}\}$ be the union-closure of $\mathcal{D}$. We now define $\hat{f}$ which is an extension of f to $\mathcal{D}^c$. If $S \in \mathcal{D}$ then $\hat{f}(S) = f(S)$. If $S \notin \mathcal{D}$ (and $S \in \mathcal{D}^c$) then $\hat{f}(S)$ is minimum value of $\sum_{i=1}^k f(S_i)$ over all partition $(S_1, \ldots, S_k)$ of S $(\cup_{i=1}^{k} S_i = S)$ such that $S_i \in \mathcal{D}$ for all $1 \leq i \leq k$. Let M be the maximum value of $\hat{f}$ on $\mathcal{D}^c$. We define an extension of $\hat{f}$ to $2^{[m]}$ by assigning value M to each set not in $\mathcal{D}^c$. Let this extension be $\tilde{f}$. We claim that $\tilde{f}$ is subadditive.

Note that M is the maximum value of $\tilde{f}$. Let A and B be any two sets. If any of A or B is not in $\mathcal{D}^c$ then $\tilde{f}(A) + \tilde{f}(B)$ is at least M and thus $\tilde{f}(A) + \tilde{f}(B) \geq \tilde{f}(A \cup B)$. Therefore, we assume both A and B are in $\mathcal{D}^c$ which implies $A \cup B$ is also in $\mathcal{D}^c$. Let A be the union of $A_1, \ldots, A_r \in \mathcal{D}$ $(r \geq 1)$ and B be the union of $B_1, \ldots, B_{r'} \in \mathcal{D}$ $(r' \geq 1)$. Therefore, $A \cup B$ is union of $A_1, \ldots, A_r, B_1, \ldots, B_{r'}$. If $A \cup B$ is in $\mathcal{D}$ then by assumption and otherwise by definition of $\hat{f}$, we have $\tilde{f}(A) + \tilde{f}(B) \geq \tilde{f}(A \cup B)$. ◀

A.2 Independence Functions for Matroids

Proof of Theorem 9. Recall the reduction from Monotone-NAE-3SAT to Matroid Extension. Given an instance of Monotone-NAE-3SAT (formula ϕ with n' variables and m' clauses), the instance of Matroid Extension is a set of defined points $\mathcal{D}$ with $|\mathcal{D}| = 4n' + 2m' + 1$ and a function h on $\mathcal{D}$. From Theorem 8, ϕ has a satisfying assignment iff h is extensible to the independence function of a graphic matroid.

Let $\mathcal{F}$ be the family of independence functions of graphic matroids. Let $\epsilon = \frac{1}{2|\mathcal{D}|}$ (and hence $\epsilon < 1/|\mathcal{D}|$) and μ be a uniform distribution over $\{(S, h(S)) | S \in \mathcal{D}\}$. Now suppose a (randomized) algorithm A PAC-learns $\mathcal{F}$ and returns a function g. If the partial function constructed is extensible, then $g(S)$ must be the independence function of a graphic matroid, and $g(S) = h(S)$ for all $S \in \mathcal{D}$ (since $\epsilon < 1/|\mathcal{D}|$ and g must satisfy $Pr_{S \sim \mu}[g(S) = h(S)] \geq 1 - \epsilon$). If the partial function is not extensible, then clearly the two conditions cannot be simultaneously satisfied. We are thus left with the problem of verifying these conditions in polynomial time, given oracle access to g. Both these conditions can be checked in polynomial time, the first by Seymour's reconstruction algorithm, and the second by simply querying $g(S)$ for all $S \in \mathcal{D}$. We can thus efficiently determine efficiently if the partial function is extensible to a graphic matroid, and hence RP $=$ NP. ◀