SIAM J. COMPUT.
Vol. 43, No. 2, pp. 514–542

MONOTONE SUBMODULAR MAXIMIZATION OVER A MATROID VIA NON-OBLIVIOUS LOCAL SEARCH*

YUVAL FILMUS† AND JUSTIN WARD‡

Abstract. We present an optimal, combinatorial $1-1/e$ approximation algorithm for monotone submodular optimization over a matroid constraint. Compared to the continuous greedy algorithm [G. Calinescu et al., *IPCO*, Springer, Berlin, 2007, pp. 182–196] our algorithm is extremely simple and requires no rounding. It consists of the greedy algorithm followed by a local search. Both phases are run not on the actual objective function, but on a related auxiliary potential function, which is also monotone and submodular. In our previous work on maximum coverage [Y. Filmus and J. Ward, *FOCS*, IEEE, Piscataway, NJ, 2012, pp. 659–668], the potential function gives more weight to elements covered multiple times. We generalize this approach from coverage functions to arbitrary monotone submodular functions. When the objective function is a coverage function, both definitions of the potential function coincide. Our approach generalizes to the case where the monotone submodular function has restricted curvature. For any curvature c, we adapt our algorithm to produce a $(1-e^{-c})/c$ approximation. This matches results of Vondrák [*STOC*, ACM, New York, 2008, pp. 67–74], who has shown that the continuous greedy algorithm produces a $(1-e^{-c})/c$ approximation when the objective function has curvature c with respect to the optimum, and proved that achieving any better approximation ratio is impossible in the value oracle model.

Key words. approximation algorithms, submodular functions, matroids, local search

AMS subject classification. 68W25

DOI. 10.1137/130920277

1. Introduction.

In this paper, we consider the problem of maximizing a monotone submodular function f, subject to a single matroid constraint. Formally, let $\mathcal{U}$ be a set of n elements and let $f\colon 2^{\mathcal{U}} \to \mathbb{R}$ be a function assigning a value to each subset of $\mathcal{U}$. We say that f is *submodular* if

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$$

for all $A, B \subseteq \mathcal{U}$. If additionally f is *monotone*, that is $f(A) \leq f(B)$ whenever $A \subseteq B$, we say that f is *monotone submodular*. Submodular functions exhibit (and are, in fact, alternately characterized by) the property of diminishing returns—if f is submodular, then $f(A \cup \{x\}) - f(A) \leq f(B \cup \{x\}) - f(B)$ for all $B \subseteq A$. Hence, they are useful for modeling economic and game-theoretic scenarios, as well as various combinatorial problems. In a general monotone submodular maximization problem, we are given a value oracle for f and a membership oracle for some distinguished collection $\mathcal{I} \subseteq 2^{\mathcal{U}}$ of *feasible sets*, and our goal is to find a member of $\mathcal{I}$ that maximizes the value of f. We assume further that f is *normalized* so that $f(\emptyset) = 0$.

We consider the restricted setting in which the collection $\mathcal{I}$ forms a matroid. Matroids are intimately connected to combinatorial optimization: the problem of optimizing a linear function over a hereditary set system (a set system closed under taking subsets) is solved optimally for all possible functions by the standard greedy algorithm if and only if the set system is a matroid [32, 11].

*Received by the editors May 8, 2013; accepted for publication (in revised form) January 14, 2014; published electronically March 27, 2014. A preliminary version of this paper appeared in [18].
http://www.siam.org/journals/sicomp/43-2/92027.html

†Institute for Advanced Study, School of Mathematics, Princeton, NJ (yfilmus@ias.edu).

‡Department of Computer Science, University of Warwick, Coventry, CV4 7AL, United Kingdom (J.D.Ward@dcs.warwick.ac.uk). This author's work was partially supported by EPSRC grant EP/J021814/1.

In the case of a *monotone submodular* objective function, the standard greedy algorithm, which takes at each step the element yielding the largest increase in f while maintaining independence, is (only) a 1/2-approximation [20]. Recently, Calinescu et al. [7, 8] and Vondrák [34] have developed a $(1-1/e)$-approximation for this problem via the *continuous greedy algorithm*, which is reminiscent of the classical Frank–Wolfe algorithm [21], producing a fractional solution. The fractional solution is rounded using pipage rounding [1] or swap rounding [10]. Recently, a fast variant of this algorithm running in time $\tilde{O}(n^2)$ has been designed by Ashwinkumar and Vondrák [3].

Feige [12] has shown that improving the bound $(1-1/e)$ is NP-hard even if f is an explicitly given coverage function (the objective function of an instance of *maximum coverage*). Nemhauser and Wolsey [30] have shown that any improvement over $(1-1/e)$ requires an exponential number of queries in the value oracle setting.

Following Vondrák [35], we also consider the case when f has restricted curvature. We say that f has *curvature* c if for any two disjoint $A, B \subseteq \mathcal{U}$,

$$f(A \cup B) \geq f(A) + (1-c)f(B).$$

When $c = 1$, this is a restatement of monotonicity of f, and when $c = 0$, linearity of f. Vondrák [35] has shown that the continuous greedy algorithm produces a $(1-e^{-c})/c$ approximation when f has curvature c. In fact, he shows that this is true even for the weaker definition of *curvature with respect to the optimum*. Furthermore, for this weaker notion of curvature, he has shown that any improvement over $(1-e^{-c})/c$ requires an exponential number of queries in the value oracle setting. The optimal approximation ratio for functions of *unrestricted* curvature c has recently been determined to be $(1-c/e)$ by Sviridenko and Ward [33], who use the non-oblivious local search approach described in this paper.

1.1. Our contribution. In this paper, we propose a conceptually simple randomized polynomial time local search algorithm for the problem of monotone submodular matroid maximization. Like the continuous greedy algorithm, our algorithm delivers the optimal $(1-1/e)$-approximation. However, unlike the continuous greedy algorithm, our algorithm is entirely combinatorial, in the sense that it deals only with integral solutions to the problem and hence involves no rounding procedure. As such, we believe that the algorithm may serve as a gateway to further improved algorithms in contexts where pipage rounding and swap rounding break down, such as submodular maximization subject to multiple matroid constraints. Its combinatorial nature has another advantage: the algorithm only evaluates the objective function on independent sets of the matroid.

Our main results are a combinatorial $1-1/e-\epsilon$ approximation algorithm for monotone submodular matroid maximization, running in time $\tilde{O}(\epsilon^{-3}r^4n)$, and a combinatorial $1-1/e$ approximation algorithm running in time $\tilde{O}(r^7n^2)$, where r is the rank of the given matroid and n is the size of its ground set. Both algorithms are randomized, and succeed with probability $1-1/n$. Our algorithm further generalizes to the case in which the submodular function has curvature c *with respect to the optimum* (see section 2 for a definition). In this case the approximation ratios obtained are $(1-e^{-c})/c-\epsilon$ and $(1-e^{-c})/c$, respectively, again matching the performance of the continuous greedy algorithm [35]. Unlike the continuous greedy algorithm, our algorithm requires knowledge of c. However, by enumerating over values of c we are able to obtain a combinatorial $(1-e^{-c})/c$ algorithm even in the case that f's curvature is unknown.[1]

[1]For technical reasons, we require that f has curvature bounded away from zero in this case.

Our algorithmic approach is based on local search. In classical local search, the algorithm starts at an arbitrary solution, and proceeds by iteratively making small changes that improve the objective function, until no such improvement can be made. A natural, worst-case guarantee on the approximation performance of a local search algorithm is the *locality ratio*, given as $\min f(S)/f(O)$, where S is a locally optimal solution (i.e., a solution which cannot be improved by the small changes considered by the algorithm), O is a global optimum, and f is the objective function.

In many cases, classical local search may have a suboptimal locality ratio, implying that a locally optimal solution may be of significantly lower quality than the global optimum. For example, for monotone submodular maximization over a matroid, the locality ratio for an algorithm changing a single element at each step is $1/2$ [20]. *Non-oblivious* local search, a technique first proposed by Alimonti [2] and by Khanna et al. [26], attempts to avoid this problem by making use of a secondary potential function to guide the search. By carefully choosing this auxiliary function, we ensure that poor local optima with respect to the original objective function are no longer local optima with respect to the new potential function. This is the approach that we adopt in the design of our local search algorithm. Specifically, we consider a simple local search algorithm in which the value of a solution is measured with respect to a carefully designed potential function g, rather than the submodular objective function f. We show that solutions which are locally optimal with respect to g have significantly higher worst-case quality (as measured by the problem's original potential function f) than those which are locally optimal with respect to f.

In our previous work [17], we designed an optimal non-oblivious local search algorithm for the restricted case of maximum coverage subject to a matroid constraint. In this problem, we are given a weighted universe of elements, a collection of sets, and a matroid defined on this collection. The goal is to find a collection of sets that is independent in the matroid and covers elements of maximum total weight. The non-oblivious potential function used in [17] gives extra weight to solutions that cover elements multiple times. That is, the potential function depends critically on the coverage representation of the objective function. In the present work, we extend this approach to *general* monotone submodular functions. This presents two challenges: defining a non-oblivious potential function without referencing the coverage representation, and analyzing the resulting algorithm.

In order to define the general potential function, we construct a generalized variant of the potential function from [17] that does not require a coverage representation. Instead, the potential function aggregates information obtained by applying the objective function to all subsets of the input, weighted according to their size. Intuitively, the resulting potential function gives extra weight to solutions that contain a large number of good subsolutions or, equivalently, remain good solutions, in expectation, when elements are removed by a random process. An appropriate setting of the weights defining our potential function yields a function which coincides with the previous definition for coverage functions, but still makes sense for arbitrary monotone submodular functions.

The analysis of the algorithm in [17] is relatively straightforward. For each type of element in the universe of the coverage problem, we must prove a certain inequality among the coefficients defining the potential function. In the general setting, however, we need to construct a proof using only the inequalities given by monotonicity and submodularity. The resulting proof is nonobvious and delicate.

This paper extends and simplifies a previous work by the same authors. The paper [18], appearing in FOCS 2012, only discusses the case $c = 1$. The general

case is discussed in [19], which is contained in arXiv. The potential functions used to guide the non-oblivious local search in both the unrestricted curvature case [18] and the maximum coverage case [17] are special cases of the function g we discuss in the present paper.[2] An exposition of the ideas of both [17] and [19] can be found in the second author's thesis [37]. In particular, the thesis explains how the auxiliary objective function can be determined by solving a linear program, both in the special case of maximum coverage and in the general case of monotone submodular functions with restricted curvature.

1.2. Related work. Fisher, Nemhauser, and Wolsey [31, 20] analyze greedy and local search algorithms for submodular maximization subject to various constraints, including single and multiple matroid constraints. They obtain some of the earliest results in the area, including a $1/(k+1)$-approximation algorithm for monotone submodular maximization subject to k matroid constraints. A recent survey by Goundan and Schulz [24] reviews many results pertaining to the greedy algorithm for submodular maximization.

More recently, Lee, Sviridenko, and Vondrák [29] consider the problem of both monotone and non-monotone submodular maximization subject to multiple matroid constraints, attaining a $1/(k+\epsilon)$-approximation for monotone submodular maximization subject to $k \geq 2$ matroid constraints using a local search. Feldman et al. [16] show that a local search algorithm attains the same bound for the related class of k-exchange systems, which includes the intersection of k strongly base orderable matroids, as well as the independent set problem in $(k+1)$-claw free graphs. Further work by Ward [36] shows that a non-oblivious local search routine attains an improved approximation ratio of $2/(k+3) - \epsilon$ for this class of problems.

In the case of unconstrained non-monotone maximization, Feige, Mirrokni, and Vondrák [13] give a 2/5-approximation algorithm via a randomized local search algorithm, and give an upper bound of 1/2 in the value oracle model. Gharan and Vondrák [22] improved the algorithmic result to 0.41 by enhancing the local search algorithm with ideas borrowed from simulated annealing. Feldman, Naor, and Schwarz [15] later improved this to 0.42 by using a variant of the continuous greedy algorithm. Buchbinder et al. have recently obtained an optimal 1/2-approximation algorithm [5].

In the setting of constrained non-monotone submodular maximization, Lee et al. [28] give a $1/(k+2+\frac{1}{k}+\epsilon)$-approximation algorithm for the case of k matroid constraints and a $(1/5-\epsilon)$-approximation algorithm for k knapsack constraints. Further work by Lee, Sviridenko, and Vondrák [29] improves the approximation ratio in the case of k matroid constraints to $1/(k+1+\frac{1}{k-1}+\epsilon)$. Feldman et al. [16] attain this ratio for k-exchange systems. Chekuri, Vondrák, and Zenklusen [9] present a general framework for optimizing submodular functions over downward-closed families of sets. Their approach combines several algorithms for optimizing the multilinear relaxation along with dependent randomized rounding via *contention resolution schemes*. As an application, they provide constant-factor approximation algorithms for several submodular maximization problems.

In the case of non-monotone submodular maximization subject to a *single* matroid constraint, Feldman, Naor, and Schwarz [14] show that a version of the continuous greedy algorithm attains an approximation ratio of $1/e$. They additionally unify various applications of the continuous greedy algorithm and obtain improved approx-

[2] The functions from [18, 19] are defined in terms of certain coefficients γ, which depend on a parameter E. Our definition here corresponds to the choice $E = e^c$. We examine the case of coverage functions in more detail in section 8.3.

imations for non-monotone submodular maximization subject to a matroid constraint or $O(1)$ knapsack constraints. Buchbinder et al. [6] further improve the approximation ratio for non-monotone submodular maximization subject to a cardinality constraint to $1/e + 0.004$, and present a 0.356-approximation algorithm for non-monotone submodular maximization subject to an exact cardinality constraint. They also present fast algorithms for these problems with slightly worse approximation ratios.

1.3. Organization of the paper. We begin by giving some basic definitions in section 2. In section 3 we introduce our basic, non-oblivious local search algorithm, which makes use of an auxiliary potential function g. In section 4, we give the formal definition of g, together with several of its properties. Unfortunately, exact computation of the function g requires evaluating f on an exponential number of sets. In section 5 we present a simplified analysis of our algorithm, under the assumption that an oracle for computing the function g is given. In section 6 we explain how we constructed the function g. In section 7 we then show how to remove this assumption to obtain our main, randomized polynomial time algorithm. The resulting algorithm uses a polynomial-time random sampling procedure to compute the function g *approximately*. Finally, some simple extensions of our algorithm are described in section 8.

2. Definitions.

Notation. If B is some Boolean condition, then

$$[\![B]\!] = \begin{cases} 1 & \text{if } B \text{ is true,} \\ 0 & \text{if } B \text{ is false.} \end{cases}$$

For a natural number n, $[n] = \{1, \ldots, n\}$. We use H_k to denote the kth Harmonic number,

$$H_k = \sum_{t=1}^{k} \frac{1}{t}.$$

It is well known that $H_k = \Theta(\ln k)$, where $\ln k$ is the natural logarithm.

For a set S and an element x, we use the shorthands $S + x = S \cup \{x\}$ and $S - x = S \setminus \{x\}$. We use the notation $S + x$ even when $x \in S$, in which case $S + x = S$, and the notation $S - x$ even when $x \notin S$, in which case $S - x = S$.

Let $\mathcal{U}$ be a set. A *set-function* f on $\mathcal{U}$ is a function $f \colon 2^{\mathcal{U}} \to \mathbb{R}$ whose arguments are subsets of $\mathcal{U}$. For $x \in \mathcal{U}$, we use $f(x) = f(\{x\})$. For $A, B \subseteq \mathcal{U}$, the *marginal value* of B with respect to A is

$$f_A(B) = f(A \cup B) - f(A).$$

Properties of set-functions. A set-function f is *normalized* if $f(\emptyset) = 0$. It is *monotone* if whenever $A \subseteq B$, then $f(A) \leq f(B)$. It is *submodular* if whenever $A \subseteq B$ and C is disjoint from B, $f_A(C) \geq f_B(C)$. If f is monotone, we need not assume that B and C are disjoint. Submodularity is equivalently characterized by the inequality

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$$

for all A and B.

The set-function f has *total curvature* c if for all $A \subseteq \mathcal{U}$ and $x \notin A$, $f_A(x) \geq (1-c)f(x)$. Equivalently, $f_A(B) \geq (1-c)f(B)$ for all disjoint $A, B \subseteq \mathcal{U}$. Note that if f has curvature c and $c' \geq c$, then f also has curvature c'. Every monotone function thus has curvature 1. A function with curvature 0 is linear; that is, $f_A(x) = f(x)$.

Following [35] we shall consider the more general notion curvature of a function *with respect to some set* $B \subseteq \mathcal{U}$. We say that f has curvature at most c with respect to a set B if

$$f(A \cup B) - f(B) + \sum_{x \in A \cap B} f_{A \cup B - x}(x) \geq (1-c)f(A) \tag{2.1}$$

for all sets $A \subseteq \mathcal{U}$. As shown in [35], if a submodular function f has total curvature at most c, then it has curvature at most c with respect to every set $A \subseteq \mathcal{U}$.

Matroids. A matroid $\mathcal{M} = (\mathcal{U}, \mathcal{I})$ is composed of a ground set $\mathcal{U}$ and a nonempty collection $\mathcal{I}$ of subsets of $\mathcal{U}$ satisfying the following two properties: (1) if $A \in \mathcal{I}$ and $B \subseteq A$, then $B \in \mathcal{I}$; (2) if $A, B \in \mathcal{I}$ and $|A| > |B|$, then $B + x \in \mathcal{I}$ for some $x \in A \setminus B$.

The sets in $\mathcal{I}$ are called *independent sets*. Maximal independent sets are known as *bases*. Condition (2) implies that all bases of the matroid have the same size. This common size is called the *rank* of the matroid.

One simple example is a *partition matroid*. The universe $\mathcal{U}$ is partitioned into r parts $\mathcal{U}_1, \ldots, \mathcal{U}_r$, and a set is independent if it contains at most one element from each part.

If A is an independent set, then the *contracted matroid* $\mathcal{M}/A = (\mathcal{U} \setminus A, \mathcal{I}/A)$ is given by

$$\mathcal{I}/A = \{B \subseteq \mathcal{U} \setminus A : A \cup B \in \mathcal{M}\}.$$

Monotone submodular maximization over a matroid. An instance of *monotone submodular maximization over a matroid* is given by $(\mathcal{M} = (\mathcal{U}, \mathcal{I}), f)$, where $\mathcal{M}$ is a matroid and f is a set-function on $\mathcal{U}$ which is normalized, monotone, and submodular.

The *optimum* of the instance is

$$f^* = \max_{O \in \mathcal{I}} f(O).$$

Because f is monotone, the maximum is always attained at some basis.

We say that a set $S \in \mathcal{I}$ is an α-*approximate* solution if $f(S) \geq \alpha f(O)$. Thus $0 \leq \alpha \leq 1$. We say that an algorithm has an *approximation ratio* of α (or, simply, that an algorithm *provides an* α-*approximation*) if it produces an α-approximate solution on every instance.

3. The algorithm. Our non-oblivious local search algorithm is shown in Algorithm 1. The algorithm takes the following input parameters.

(i) A matroid $\mathcal{M} = (\mathcal{U}, \mathcal{I})$, given as a ground set $\mathcal{U}$ and a membership oracle for some collection $\mathcal{I} \subseteq 2^{\mathcal{U}}$ of independent sets, which returns whether or not $X \in \mathcal{I}$ for any $X \subseteq \mathcal{U}$.

(ii) A monotone submodular function $f\colon 2^{\mathcal{U}} \to \mathbb{R}_{\geq 0}$, given as a value oracle that returns $f(X)$ for any $X \subseteq \mathcal{U}$.

(iii) An upper bound $c \in (0, 1]$ on the curvature of f. The case in which the curvature of f is unrestricted corresponds to $c = 1$.

(iv) A convergence parameter ϵ.

Throughout the paper, we let r denote the rank of $\mathcal{M}$ and $n = |\mathcal{U}|$.

Algorithm 1. The non-oblivious local search algorithm.

Input: $\mathcal{M} = (\mathcal{U}, \mathcal{I})$, f, c, ϵ
Set $\epsilon_1 = \frac{\epsilon}{rH_r}$
Let S_{init} be the result of running the standard greedy algorithm on $(\mathcal{M}, g)$
$S \leftarrow S_{\text{init}}$
repeat
  foreach *element* $x \in S$ *and* $y \in \mathcal{U} \setminus S$ **do**
    $S' \leftarrow S - x + y$
    if $S' \in \mathcal{I}$ *and* $g(S') > (1+\epsilon_1)g(S)$ **then** {`An improved solution` S' `was found`}
      $S \leftarrow S'$ {`update the current solution`}
      break {`and continue to the next iteration`}
until *No exchange is made*
return S

The algorithm starts from an initial greedy solution S_{init}, and proceeds by repeatedly exchanging one element x in the current solution S for one element y not in S, with the aim of obtaining an improved independent set $S' \in \mathcal{I}$. In both the initial greedy phase and the following local search phase, the quality of the solution is measured not with respect to f, but rather with respect to an auxiliary potential function g (as we discuss shortly, we in fact must use an estimate $\tilde{g}$ for g), which is determined by the rank of $\mathcal{M}$ and the value of the curvature bound c.

We give a full definition of g in section 4. The function is determined by a sequence of coefficients depending on the upper bound c on the curvature of f. Evaluating the function g exactly will require an exponential number of value queries to f. Nonetheless, in section 7 we show how to modify Algorithm 1 by using a random sampling procedure to approximate g. The resulting algorithm has the desired approximation guarantee with high probability and runs in polynomial time.

At each step we require that an improvement increase g by a factor of at least $1+\epsilon_1$. This, together with the initial greedy choice of S_{init}, ensures that Algorithm 1 converges in time polynomial in r and n, at the cost of a slight loss in its locality gap. In section 8 we describe how the small resulting loss in the approximation ratio can be recovered, both in the case of Algorithm 1, and in the randomized, polynomial-time variant we consider in section 7.

The greedy phase of Algorithm 1 can be replaced by simpler phases, at the cost of a small increase in the running time. The number of iterations of Algorithm 1 can be bounded by $\log_{1+\epsilon_1} \frac{g^*}{g(S_{\text{init}})}$, where $g^* = \max_{A \in \mathcal{I}} g(A)$. When S_{init} is obtained as in Algorithm 1, $g(S_{\text{init}}) \geq g^*/2$ and so the number of iterations is at most $\log_{1+\epsilon_1} 2 = O(\epsilon_1^{-1})$. If instead we generate S_{init} by running the greedy algorithm on f, Lemma 4.4 below shows that $g(S_{\text{init}}) \geq f(S_{\text{init}}) \geq f^*/2 \geq \Omega(g^*/\log r)$, where $f^* = \max_{A \in \mathcal{I}} f(A)$. Hence the number of iterations is $O(\epsilon_1^{-1} \log\log r)$, and so the resulting algorithm is slower by a multiplicative factor of $O(\log\log r)$. An even simpler initialization phase finds $x^* = \operatorname{argmax}_{x \in \mathcal{U}} f(x)$, and completes it to a set $S_{\text{init}} \in \mathcal{I}$ arbitrarily. Such a set satisfies $f(S_{\text{init}}) \geq f^*/r = \Omega(g^*/r\log r)$, hence the number of iterations is $O(\epsilon_1^{-1} \log r)$, resulting in an algorithm slower than Algorithm 1 by a multiplicative factor of $O(\log r)$.

4. The auxiliary objective function g. We turn to the remaining task needed for completing the definition of Algorithm 1: giving a definition of the potential

function g. The construction we use for g will necessarily depend on c, but because we have fixed an instance, we shall omit this dependence from our notation, in order to avoid clutter. We also assume throughout that the function f is normalized ($f(\emptyset) = 0$).

4.1. Definition of g. We now present a definition of our auxiliary potential function g. Our goal is to give extra value to solutions S that are robust with respect to small changes. That is, we would like our potential function to assign higher value to solutions that retain their quality even when some of their elements are removed by future iterations of the local search algorithm. We model this general notion of robustness by considering a random process that obtains a new solution T from the current solution S by independently discarding each element of S with some probability. Then we use the expected value of $f(T)$ to define our potential function g.

It will be somewhat more intuitive to begin by relating the *marginals* g_A of g to the *marginals* f_A of f, rather than directly defining the values of g and f. We begin by considering some simple properties that we would like to hold for the marginals, and eventually give a concrete definition of g, showing that it has these properties.

Let A be some subset of $\mathcal{U}$ and consider an element $x \notin A$. We want to define the marginal value $g_A(x)$. We consider a two-step random process that first selects a probability p from an appropriate continuous distribution, then a set $B \subseteq A$ by choosing each element of A independently with some probability p. We then define g so that $g_A(x)$ is the expected value of $f_B(x)$ over the random choice of B.

Formally, let P be a continuous distribution supported on $[0, 1]$ with density given by $ce^{cx}/(e^c - 1)$. Then, for each $A \subseteq \mathcal{U}$, we consider the probability distribution μ_A on 2^A given by

$$\mu_A(B) = \mathop{\mathbb{E}}_{p\sim P} p^{|B|}(1-p)^{|A|-|B|}.$$

Note that this is simply the expectation over our initial choice of p of the probability that the set B is obtained from A by randomly selecting each element of A independently with probability p. Furthermore, for any A and any $A' \subseteq A$, if $B \sim \mu_A$, then $B \cap A' \sim \mu_{A'}$.

Given the distributions μ_A, we shall construct a function g so that

$$g_A(x) = \mathop{\mathbb{E}}_{B\sim\mu_A} [f_B(x)]. \tag{4.1}$$

That is, the marginal value $g_A(x)$ is the expected marginal gain in f obtained when x is added to a random subset of A, obtained by the two-step experiment we have just described.

We can obtain some further intuition by considering how the distribution P affects the values defined in (4.1). In the extreme example in which $p = 1$ with probability 1, we have $g_A(x) = f_A(x)$ and so g behaves exactly like the original submodular function. Similarly, if $p = 0$ with probability 1, then $g_A(x) = f_\emptyset(x) = f(\{x\})$ for all A, and so g is in fact a linear function. Thus, we can intuitively think of the distribution P as blending together the original function f with some other "more linear" approximations of f, which have systematically reduced curvature. We shall see that our choice of distribution results in a function g that gives the desired locality gap.

It remains to show that it is possible to construct a function g whose marginals satisfy (4.1). In order to do this, we first note that the probability $\mu_A(B)$ depends

only on $|A|$ and $|B|$. Thus, if we define the values

$$m_{a,b} = \mathop{\mathbb{E}}_{p\sim P} p^b(1-p)^{a-b} = \int_0^1 \frac{ce^{cp}}{e^c-1}\cdot p^b(1-p)^{a-b}\,dp$$

for all $a \geq b \geq 0$, then we have $\mu_A(B) = m_{|A|,|B|}$. We adopt the convention that $m_{a,b} = 0$ if either a or b is negative. Then, we consider the function g given by

$$g(A) = \sum_{B\subseteq A} m_{|A|-1,|B|-1} f(B). \tag{4.2}$$

The marginals of this function are given by

$$\begin{aligned} g_A(x) &= g(A+x) - g(A) \\ &= \sum_{B\subseteq A+x} m_{|A|,|B|-1} f(B) - \sum_{B\subseteq A} m_{|A|-1,|B|-1} f(B) \\ &= \sum_{B\subseteq A} \left(m_{|A|,|B|-1} - m_{|A|-1,|B|-1}\right) f(B) + m_{|A|,|B|} f(B+x). \end{aligned}$$

When $b > 0$, the term $m_{a,b-1} - m_{a-1,b-1}$ evaluates to

$$\begin{aligned} m_{a,b-1} - m_{a-1,b-1} &= \mathop{\mathbb{E}}_{p\sim P}[p^{b-1}(1-p)^{a-b+1} - p^{b-1}(1-p)^{a-b}] \\ &= \mathop{\mathbb{E}}_{p\sim P}[-p^b(1-p)^{a-b}] \\ &= -m_{a,b}. \end{aligned}$$

Since f is normalized, we trivially have $(m_{|A|,-1} - m_{|A|-1,-1})f(\emptyset) = -m_{|A|,0} f(\emptyset)$. We conclude that

$$\begin{aligned} g_A(x) &= \sum_{B\subseteq A} -m_{|A|,|B|} f(B) + m_{|A|,|B|} f(B+x) \\ &= \sum_{B\subseteq A} m_{|A|,|B|} f_B(x) \\ &= \mathop{\mathbb{E}}_{B\sim\mu_A}[f_B(x)]. \end{aligned}$$

The values $m_{a,b}$ used to define g in (4.2) can be computed from the following recurrence, which will also play a role in our analysis of the locality gap of Algorithm 1.

LEMMA 4.1. *$m_{0,0} = 1$, and for $a > 0$ and $0 \leq b \leq a$,*

$$cm_{a,b} = (a-b)m_{a-1,b} - bm_{a-1,b-1} + \begin{cases} -c/(e^c-1) & \text{if } b = 0, \\ 0 & \text{if } 0 < a < b, \\ ce^c/(e^c-1) & \text{if } a = b. \end{cases}$$

Proof. For the base case, we have

$$m_{0,0} = \int_0^1 \frac{ce^{cp}}{e^c-1}\,dp = 1.$$

The proof of the general case follows from a simple integration by parts:

$$\begin{aligned} cm_{a,b} &= c\int_0^1 \frac{ce^{cp}}{e^c-1}\cdot p^b(1-p)^{a-b}\,dp \\ &= c\cdot\frac{e^{cp}}{e^c-1}\cdot p^b(1-p)^{a-b}\Big|_{p=0}^{p=1} \\ &\quad - c\int_0^1 \left[bp^{b-1}(1-p)^{a-b} - (a-b)p^b(1-p)^{a-b-1}\right]\frac{e^{cp}}{e^c-1}\,dp \\ &= \frac{[\![a=b]\!]ce^c - [\![b=0]\!]c}{e^c-1} + (a-b)m_{a-1,b} - bm_{a-1,b-1}. \end{aligned}$$

(When $b=0$ the integrand simplifies to $\frac{ce^{cp}}{e^c-1}\cdot(1-p)^a$, and when $a=b$ it simplifies to $\frac{ce^{cp}}{e^c-1}\cdot p^b$.) □

In future proofs, we shall also need the following upper bound on the sum of the coefficients appearing in (4.2). Define

$$\tau(A) = \sum_{B\subseteq A} m_{|A|-1,|B|-1}.$$

The quantity $\tau(A)$ depends only on $|A|$, so we can define a sequence ℓ_k by $\tau(A) = \ell_{|A|}$. This sequence is given by the following formula and recurrence.

LEMMA 4.2. *ℓ_k is given by the formula*

$$\ell_k = \int_0^1 \frac{ce^{cp}}{e^c-1}\cdot\frac{1-(1-p)^k}{p}\,dp$$

and by the recurrence

$$\ell_0 = 0, \quad \ell_{k+1} = \ell_k + m_{k,0}.$$

Furthermore,

$$\ell_k \le \frac{ce^c}{e^c-1}H_k.$$

Proof. The formula for ℓ_k follows directly from the formula for $m_{a,b}$ together with the binomial formula:

$$\begin{aligned} \ell_k &= \sum_{t=1}^k \binom{k}{t} m_{k-1,t-1} \\ &= \int_0^1 \frac{ce^{cp}}{e^c-1}\cdot\sum_{t=1}^k\binom{k}{t}p^{t-1}(1-p)^{k-t}\,dp \\ &= \int_0^1 \frac{ce^{cp}}{e^c-1}\cdot\frac{1-(1-p)^k}{p}\,dp. \end{aligned}$$

This formula allows us to bound ℓ_k:

$$\begin{aligned}\ell_k &\leq \frac{ce^c}{e^c-1}\int_0^1 \frac{1-(1-p)^k}{p}\,dp \\ &= \frac{ce^c}{e^c-1}\int_0^1 \sum_{t=0}^{k-1}(1-p)^t\,dp \\ &= \frac{ce^c}{e^c-1}\sum_{t=0}^{k-1}\frac{1}{t+1} = \frac{ce^c}{e^c-1}H_k.\end{aligned}$$

Clearly $\ell_0 = 0$. The recurrence follows by calculating $\ell_{k+1} - \ell_k$:

$$\begin{aligned}\ell_{k+1}-\ell_k &= \int_0^1 \frac{ce^{cp}}{e^c-1}\cdot\left[\frac{1-(1-p)^{k+1}}{p} - \frac{1-(1-p)^k}{p}\right]dp \\ &= \int_0^1 \frac{ce^{cp}}{e^c-1}\cdot(1-p)^k\,dp = m_{k,0}. \qquad \square\end{aligned}$$

We thank an anonymous reviewer for simplifying the proof of the upper bound. For an alternative proof of the formula and the recurrence, see section 8.3.

4.2. Properties of g. We now show that our potential function g shares many basic properties with f.

LEMMA 4.3. *The function g is normalized, monotone, submodular, and has curvature at most c.*

Proof. From (4.2) we have $g(\emptyset) = m_{-1,-1}f(\emptyset) = 0$. Thus, g is normalized. Additionally, (4.1) immediately implies that g is monotone, since the monotonicity of f implies that each term $f_B(x)$ is nonnegative. Next, suppose that $A_1 \subseteq A_2$ and $x \notin A_2$. Then from (4.1), we have

$$g_{A_2}(x) = \mathop{\mathbb{E}}_{B\sim\mu_{A_2}} f_B(x) \leq \mathop{\mathbb{E}}_{B\sim\mu_{A_2}} f_{B\cap A_1}(x) = \mathop{\mathbb{E}}_{B\sim\mu_{A_1}} f_B(x) = g_{A_1}(x),$$

where the inequality follows from submodularity of f. Thus, g is submodular. Finally, for any set $A \subseteq \mathcal{U}$ and any element $x \notin A$, we have

$$g_A(x) = \mathop{\mathbb{E}}_{B\sim\mu_A} f_B(x) \geq (1-c)f(x) = (1-c)g(x),$$

where the inequality follows from the bound on the curvature of f, and the second equality from setting $A = \emptyset$ in (4.1). Thus, g has curvature at most c. In fact, it is possible to show that for any given $|A|$, g has a slightly lower curvature than f, corresponding to our intuition that the distribution P blends together f and various functions of reduced curvature. For our purposes, however, an upper bound of c is sufficient. $\square$

Finally, we note that for any $S \subseteq \mathcal{U}$, it is possible to bound the value $g(S)$ relative to $f(S)$.

LEMMA 4.4. *For any $A \subseteq \mathcal{U}$,*

$$f(A) \leq g(A) \leq \frac{ce^c}{e^c-1}H_{|A|}f(A).$$

Proof. Let $A = \{a_1, \ldots, a_{|A|}\}$ and define $A_i = \{a_1, \ldots, a_i\}$ for $0 \leq i \leq |A|$. The formula (4.1) implies that

$$g_{A_i}(a_{i+1}) = \mathop{\mathbb{E}}_{B\sim\mu_{A_i}} f_B(a_{i+1}) \geq f_{A_i}(a_{i+1}).$$

Summing the resulting inequalities for $i = 0$ to $|A| - 1$, we get

$$g(A) - g(\emptyset) \geq f(A) - f(\emptyset).$$

The lower bound then follows from the fact that both g and f are normalized, so $g(\emptyset) = f(\emptyset) = 0$.

For the upper bound, (4.2) and monotonicity of f imply that

$$g(A) = \sum_{B \subseteq A} m_{|A|-1,|B|-1} f(B) \leq f(A) \sum_{B \subseteq A} m_{|A|-1,|B|-1}.$$

The upper bound then follows directly from applying the bound of Lemma 4.2 to the final sum. □

4.3. Approximating g via sampling. Evaluating $g(A)$ exactly requires evaluating f on all subsets $B \subseteq A$, and so we cannot compute g directly without using an exponential number of calls to the value oracle f. We now show that we can efficiently estimate $g(A)$ by using a sampling procedure that requires evaluating f on only a polynomial number of sets $B \subseteq A$. In section 7, we show how to use this sampling procedure to obtain a randomized variant of Algorithm 1 that runs in polynomial time.

We have already shown how to construct the function g, and how to interpret the marginals of g as the expected value of a certain random experiment. Now we show that the direct definition of $g(A)$ in (4.2) can also be viewed as the result of a random experiment.

For a set A, consider the distribution ν_A on 2^A given by

$$\nu_A(B) = \frac{m_{|A|-1,|B|-1}}{\tau(A)}.$$

Then, recalling the direct definition of g, we have:

$$g(A) = \sum_{B \subseteq A} m_{|A|-1,|B|-1} f(B) = \tau(A) \mathop{\mathbb{E}}_{B \sim \nu_A} [f(B)].$$

We can estimate $g(A)$ to any desired accuracy by sampling from the distribution ν_A. This can be done efficiently using the recurrences for $m_{a,b}$ and $\tau(A)$ given by Lemmas 4.1 and 4.2, respectively. Let $B_1, \ldots, B_N$ be N independent random samples from ν_A. Then, we define

$$\tilde{g}(A) = \tau(A) \frac{1}{N} \sum_{i=1}^{N} f(B_i). \tag{4.3}$$

LEMMA 4.5. *Choose $M, \epsilon > 0$, and set*

$$N = \frac{1}{2} \left(\frac{ce^c}{e^c - 1} \cdot \frac{H_n}{\epsilon} \right)^2 \ln M.$$

Then,

$$\Pr[|\tilde{g}(A) - g(S)| \geq \epsilon g(S)] = O\left(M^{-1}\right).$$

Proof. We use the following version of Hoeffding's bound.

FACT 1 (Hoeffding's bound). *Let $X_1, \ldots, X_N$ be independently and identically distributed nonnegative random variables bounded by B, and let $\overline{X}$ be their average. Suppose that $\mathbb{E}\,\overline{X} \geq \rho B$. Then, for any $\epsilon > 0$,*

$$\Pr[|\overline{X} - \mathbb{E}\,\overline{X}| \geq \epsilon\,\mathbb{E}\,\overline{X}] \leq 2\exp\left(-2\epsilon^2\rho^2 N\right).$$

Consider the random variables $X_i = \tau(A)f(B_i)$. Because f is monotone and each B_i is a subset of A, each X_i is bounded by $\tau(A)f(A)$. The average $\overline{X}$ of the values X_i satisfies

$$\mathbb{E}\,\overline{X} = g(A) \geq f(A),$$

where the inequality follows from Lemma 4.4. Thus, Hoeffding's bound implies that

$$\Pr[|\overline{X} - \mathbb{E}\,\overline{X}| \geq \epsilon\,\mathbb{E}\,\overline{X}] \leq 2\exp\left(-\frac{2\epsilon^2 N}{\tau(A)^2}\right).$$

By Lemma 4.2 we have $\tau(A) \leq \frac{ce^c}{e^c-1}H_{|A|} \leq \frac{ce^c}{e^c-1}H_n$ and so

$$2\exp\left(-\frac{2\epsilon^2 N}{\tau(A)^2}\right) \leq 2\exp\left(-\ln M\right) = O\left(M^{-1}\right). \qquad \square$$

5. Analysis of Algorithm 1. We now give a complete analysis of the runtime and approximation performance of Algorithm 1. The algorithm has two phases: a greedy phase and a local search phase. Both phases are guided by the auxiliary potential function g defined in section 4. As noted in section 4.3, we cannot, in general, evaluate g in polynomial time, though we can estimate g by sampling. However, sampling g complicates the algorithm and its analysis. We postpone such concerns until the next section, and in this section suppose that we are given a value oracle returning $g(A)$ for any set $A \subseteq \mathcal{U}$. We then show that Algorithm 1 requires only a polynomial number of calls to the oracle for g. In this way, we can present the main ideas of the proofs without a discussion of the additional parameters and proofs necessary for approximating g by sampling. In the next section we use the results of Lemma 4.5 to implement an approximate oracle for g in polynomial time, and adapt the proofs given here to obtain a randomized, polynomial time algorithm.

Consider an arbitrary input to the algorithm. Let $S = \{s_1, \ldots, s_r\}$ be the solution returned by Algorithm 1 on this instance and O be an optimal solution to this instance. It follows directly from the definition of the standard greedy algorithm and the type of exchanges considered by Algorithm 1 that S is a base. Moreover, because f is monotone, we may assume without loss of generality that O is a base, as well. We index the elements o_i of O by using the following lemma of Brualdi [4].

FACT 2 (Brualdi's lemma). *Suppose A, B are two bases in a matroid. There is a bijection $\pi\colon A \to B$ such that for all $a \in A$, $A - a + \pi(a)$ is a base. Furthermore, π is the identity on $A \cap B$.*

The main difficulty in bounding the locality ratio of Algorithm 1 is that we must bound the ratio $f(S)/f(O)$ stated in terms of f, by using only the fact that S is locally optimal with respect to g. Thus, we must somehow relate the values of $f(S)$ and $g(S)$. The following theorem relates the values of f and g on arbitrary bases of a matroid. Later, we shall apply this theorem to S and O to obtain an approximation guarantee both for Algorithm 1 and for the randomized variant presented in the next section.

THEOREM 5.1. *Let $A = \{a_1, \ldots, a_r\}$ and $B = \{b_1, \ldots, b_r\}$ be any two bases of $\mathcal{M}$, and suppose f has curvature at most c with respect to B. Further suppose that*

we index the elements of B so that $b_i = \pi(a_i)$, where $\pi\colon A \to B$ is the bijection guaranteed by Brualdi's lemma. Then,

$$\frac{ce^c}{e^c-1} f(A) \geq f(B) + \sum_{i=1}^{r} [g(A) - g(A - a_i + b_i)].$$

Proof. First, we provide some general intuition for the proof. In order to prove the theorem, we fix a current base A of $\mathcal{M}$ and some other arbitrary base B, and consider each of the individual swaps from Brualdi's lemma. Each such swap removes a single element of A and adds one element of B to the result. We consider the change in g caused by each such swap. The value of g on any given set may be $O(\log n)$ times larger than the corresponding value of f. Indeed, the value of $g(A)$ is obtained by summing appropriately weighted values of $f(A')$ over all subsets $A' \subseteq A$. However, we shall show that our definition of g ensures that when all the differences $g(A) - g(A - a_i + b_i)$ are added together, most of these values cancel. Specifically, cancellation between corresponding values $f(A')$ leave us with an expression in which all terms are bounded by $f(A)$ or $f(B)$. Specifically, we show that the sum of differences $\sum_{i=1}^{r} [g(A) - g(A - a_i + b_i)]$ reduces to a lower bound on the difference between $f(B)$ and $\frac{ce^c}{e^c-1} f(A)$.

Our proof involves two inequalities and one equation, each of which we shall prove later as a separate lemma. The inequalities will pertain to the quantity

$$\sum_{i=1}^{r} g_{A-a_i}(a_i), \tag{5.1}$$

which represents (up to scaling) the average total loss in $g(A)$ when a single element a_i is removed from A.

In Lemma 5.2, we use the basic definition and properties of g to show that the loss for each a_i is bounded by the difference $g(A) - g(A - a_i + b_i)$ and a combination of marginal values of f. Then, by the linearity of expectation, we obtain

$$\sum_{i=1}^{r} g_{A-a_i}(a_i) \geq \sum_{i=1}^{r} [g(A) - g(A - a_i + b_i)] + \mathop{\mathbb{E}}_{T \sim \mu_A} \sum_{i=1}^{r} f_{T-b_i}(b_i). \tag{5.2}$$

In Lemma 5.3 (similarly to Vondrák [35, Lemma 3.1]), we simplify the final term in (5.2), using the fact that f has curvature at most c with respect to B. The lemma relates the sum of marginals $f_{T-b_i}(b_i)$ to $f(T)$, showing that

$$\sum_{i=1}^{r} f_{T-b_i}(b_i) \geq f(B) - cf(T)$$

for any $T \subseteq A$ and $b_i \in B$. Applying the resulting inequalities in (5.2), we obtain the following bound on (5.1):

$$\sum_{i=1}^{r} g_{A-a_i}(a_i) \geq \sum_{i=1}^{r} [g(A) - g(A - a_i + b_i)] + f(B) - c \mathop{\mathbb{E}}_{T \sim \mu_A} f(T). \tag{5.3}$$

Finally, in Lemma 5.4 we reconsider (5.1), expanding the definition of g and adding the final term $c\,\mathbb{E}_{T\sim\mu_A} f(T)$ of (5.3) to the result. By exploiting the recurrence

of Lemma 4.1, we show that all terms except those involving $f(A)$ vanish from the resulting expression. Specifically, we show the following equality involving (5.1):

$$\sum_{i=1}^{r} g_{A-a_i}(a_i) + c \mathop{\mathbb{E}}_{T\sim\mu_A} f(T) = \frac{ce^c}{e^c-1} f(A). \tag{5.4}$$

Combining the equality (5.4) with the lower bound on (5.1) from (5.3) then completes the proof. □

We now prove each of the necessary lemmas.

LEMMA 5.2. *For all $i \in [r]$,*

$$g_{A-a_i}(a_i) \geq g(A) - g(A - a_i + b_i) + \mathop{\mathbb{E}}_{T\sim\mu_A} f_{T-b_i}(b_i).$$

Proof. The proof relies on the characterization of the marginals of g given in (4.1). We consider two cases: $b_i \notin A$ and $b_i \in A$. If $b_i \notin A$ then the submodularity of g implies

$$\begin{aligned} g_{A-a_i}(a_i) &\geq g_{A-a_i+b_i}(a_i) \\ &= g(A + b_i) - g(A - a_i + b_i) \\ &= g_A(b_i) + g(A) - g(A - a_i + b_i) \\ &= g(A) - g(A - a_i + b_i) + \mathop{\mathbb{E}}_{T\sim\mu_A} f_T(b_i). \end{aligned}$$

On the other hand, when $b_i \in A$, we must have $b_i = \pi(a_i) = a_i$ by the definition of π. Then,

$$\begin{aligned} g_{A-a_i}(a_i) &= \mathop{\mathbb{E}}_{T\sim\mu_{A-a_i}} f_T(a_i) \\ &= \mathop{\mathbb{E}}_{T\sim\mu_A} f_{T-a_i}(a_i) \\ &= \mathop{\mathbb{E}}_{T\sim\mu_A} f_{T-b_i}(b_i) \\ &= g(A) - g(A) + \mathop{\mathbb{E}}_{T\sim\mu_A} f_{T-b_i}(b_i) \\ &= g(A) - g(A - a_i + b_i) + \mathop{\mathbb{E}}_{T\sim\mu_A} f_{T-b_i}(b_i), \end{aligned}$$

where the second equality follows from the fact that if $T \sim \mu_A$, then $T \cap (A \setminus a_i) \sim \mu_{A-a_i}$. □

LEMMA 5.3. *For any $T \subseteq A$,*

$$\sum_{i=1}^{r} f_{T-b_i}(b_i) \geq f(B) - cf(T).$$

Proof. Our proof relies only on the submodularity and curvature of f. We have

$$\begin{aligned} \sum_{i=1}^{r} f_{T-b_i}(b_i) &= \sum_{b_i\in B\setminus T} f_T(b_i) + \sum_{b_i\in B\cap T} f_{T-b_i}(b_i) \\ &\geq f(T \cup B) - f(T) + \sum_{b_i\in B\cap T} f_{T-b_i}(b_i) \\ &\geq f(T \cup B) - f(T) + \sum_{b_i\in B\cap T} f_{T\cup B-b_i}(b_i) \\ &\geq (1-c)f(T) + f(B) - f(T) \\ &= f(B) - cf(T), \end{aligned}$$

where the first two inequalities follow from the submodularity of f and the last inequality from (2.1), since f has curvature at most c with respect to B. □

LEMMA 5.4.

$$\sum_{i=1}^{r} g_{A-a_i}(a_i) + c \mathop{\mathbb{E}}_{T\sim\mu_A} f(T) = \frac{ce^c}{e^c-1} f(A). \tag{5.5}$$

Proof. The proof relies primarily on the recurrence given in Lemma 4.1 for the values $m_{a,b}$ used to define g. From the characterization of the marginals of g given in (4.1) we have

$$g_{A-a_i}(a_i) = \mathop{\mathbb{E}}_{T\sim\mu_{A-a_i}}[f_T(a_i)] = \mathop{\mathbb{E}}_{T\sim\mu_{A-a_i}}[f(T+a_i) - f(T)].$$

Each subset $D \subseteq A$ appears in the expectation. Specifically, if $a_i \in D$, then we have the term $\mu_{A-a_i}(D-a_i)f(D)$, and if $a_i \in A \setminus D$, then we have the term $-\mu_{A-a_i}(D)f(D)$. Therefore the coefficient of $f(D)$ in the left-hand side of (5.5) is thus given by

$$\begin{aligned}&\left(\sum_{a_i\in D} \mu_{A-a_i}(D-a_i)\right) - \left(\sum_{a_i\notin D} \mu_{A-a_i}(D)\right) + c\mu_A(D)\\ &\quad = |D| m_{r-1,|D|-1} - (r-|D|)m_{r-1,|D|} + c m_{r,|D|}.\end{aligned}$$

According to the recurrence for $m_{,}$ given in Lemma 4.1, the right-hand side vanishes unless $D = \emptyset$, in which case we get $\frac{-c}{e^c-1} f(\emptyset) = 0$, or $D = A$, in which case it is $\frac{ce^c}{e^c-1} f(A)$. □

We are now ready to prove this section's main claim, which gives bounds on both the approximation ratio and complexity of Algorithm 1.

THEOREM 5.5. *Algorithm* 1 *is a* $(\frac{1-e^{-c}}{c} - \epsilon)$*-approximation algorithm, requiring at most* $O(r^2 n \epsilon^{-1} \log n)$ *evaluations of* g.

Proof. We first consider the number of evaluations of g required by Algorithm 1. The initial greedy phase requires $O(rn)$ evaluations of g, as does each iteration of the local search phase. Thus, the total number of evaluations of g required by Algorithm 1 is $O(rnI)$, where I is the number of improvements applied in the local search phase. We now derive an upper bound on I.

Let $g^* = \max_{A\in\mathcal{I}} g(A)$ be the maximum value attained by g on any independent set in $\mathcal{M}$. Algorithm 1 begins by setting S to a greedy solution S_{init}, and each time it selects an improved solution S' to replace S by, we must have

$$g(S') > (1+\epsilon_1) g(S).$$

Thus, the number of improvements that Algorithm 1 can apply is at most

$$\log_{1+\epsilon_1} \frac{g^*}{g(S_{\text{init}})}.$$

Fisher, Nemhauser, and Wolsey [20] show that the greedy algorithm is a 1/2-approximation algorithm for maximizing any monotone submodular function subject to a matroid constraint. In particular, because g is monotone submodular, as shown in Lemma 4.3, we must have

$$I \le \log_{1+\epsilon_1} \frac{g^*}{g(S_{\text{init}})} \le \log_{1+\epsilon_1} 2 = O(\epsilon_1^{-1}) = O(rH_r\epsilon^{-1}) = O(r\epsilon^{-1}\log n).$$

Next, we consider the approximation ratio of Algorithm 1. Recall that O is an optimal solution of the arbitrary instance $(\mathcal{M} = (\mathcal{U},\mathcal{I}), f)$ on which Algorithm 1 returns the solution S. We apply Theorem 5.1 to the bases S and O, indexing S and

O as in the theorem so that $S - s_i + o_i \in \mathcal{I}$ for all $i \in [r]$, to obtain

$$\frac{ce^c}{e^c - 1} f(S) \geq f(O) + \sum_{i=1}^{r} [g(S) - g(S - s_i + o_i)]. \tag{5.6}$$

Then, we note that we must have

$$g(S - s_i + o_i) \leq (1 + \epsilon_1) g(S)$$

for each value $i \in [r]$—otherwise, Algorithm 1 would have exchanged s_i for o_i rather than returning S. Summing the resulting r inequalities gives

$$\sum_{i=1}^{r} [g(S) - g(S - s_i + o_i)] \geq -r\epsilon_1 g(S).$$

Applying this and the upper bound on $g(S)$ from Lemma 4.4 to (5.6) we then obtain

$$\begin{aligned} \frac{ce^c}{e^c - 1} f(S) \geq f(O) - r\epsilon_1 g(S) &\geq f(O) - \frac{ce^c}{e^c - 1} r\epsilon_1 H_r f(S) \\ &\geq f(O) - \frac{ce^c}{e^c - 1} r\epsilon_1 H_r f(O). \end{aligned}$$

Rewriting this inequality using the definition $\epsilon_1 = \frac{\epsilon}{rH_r}$ then gives

$$f(S) \geq \left(\frac{1 - e^{-c}}{c} - \epsilon \right) f(O),$$

and so Algorithm 1 is a $(\frac{1-e^{-c}}{c} - \epsilon)$-approximation algorithm. ∎

6. How g was constructed. The definition of our potential function g might seem somewhat mysterious. In this section we try to dispel some of this mystery. First, we explain how we initially found the definition of g by solving a sequence of factor-revealing linear programs (LPs). Second, we show how the definition of g follows directly from required properties in the proof of our main technical result, Theorem 5.1.

6.1. Factor-revealing LP. The main idea behind our construction of the function g was to consider a general class of possible functions, and then optimize over this class of functions. We consider a mathematical program whose variables are given by the values defining a submodular function f and the parameters defining a related potential function g, and whose objective function is the resulting locality gap for the instance defined by f. This technique of using a *factor-revealing LP*, which gives the approximation ratio of some algorithm over a particular family of instances, appears formally in Jain et al. [25] in the context of greedy facility location algorithms, but had been applied earlier by Goemans and Kleinberg [23] to analyze an algorithm for the minimum latency problem. Here, however, we use the linear program not only to find the best possible approximation ratio for our approach, but also to determine the potential function g that yields this ratio.

The class of functions that we consider are functions of the form

$$g(A) = \sum_{B \subseteq A} G_{|A|,|B|} f(A). \tag{6.1}$$

Since we assume that f is normalized, we can take $G_{a,b} = 0$ when $b = 0$. Our local search algorithm only evaluates g at sets whose sizes are the rank r of the matroid, so we can assume that $|A| = r$. (The greedy phase evaluates g at sets of smaller size, but as we explain in section 3, this phase can be replaced by phases not evaluating g at all.) The optimal choice of coefficients $G_{r,k}$, and the resulting approximation ratio are given by the following program Π:

$$\max_{G_{r,1},\ldots,G_{r,r}} \min_{f} \frac{f(S)}{f(O)}$$

subject to (s.t.)

$g(S) \geq g(S - s_i + o_i)$ for $i \in [r]$

f is normalized and monotone submodular, and has curvature c with respect to O.

Given the values of $G_{r,1}, \ldots, G_{r,r}$, the inner program (the part starting with $\min_f f(S)/f(O)$) evaluates the approximation ratio of the resulting local search algorithm, assuming that the optimal set O and the set produced by the algorithm S are disjoint (the analysis of Algorithm 1 will have to show that the same approximation ratio is obtained even when O and S are not disjoint). The variables of the inner program are the 2^{2r} values of the function f on all subsets of $S \cup O$, where $S = \{s_1, \ldots, s_r\}$ and $O = \{o_1, \ldots, o_r\}$ are two fixed disjoint sets of cardinality r. The constraints $g(S) \geq g(S - s_i + o_i)$ state that the set S is locally optimal with respect to the function g (implicitly relying on Brualdi's lemma). These constraints are expanded into linear constraints over the variables of the program by substituting the definition of g given by (6.1).

The program Π is equivalent to an LP. In order to convert Π into an LP, we first add an additional constraint $f(O) = 1$, thus making the inner program an LP. We then dualize the inner program and obtain an LP with a maximized objective function. Folding both maximization operators, we obtain an LP Π' whose value is the same as the value of Π. Since the inner program in Π has exponentially many variables, the LP Π' has exponentially many constraints. Symmetry considerations allow us to reduce the number of variables in the inner program to $O(r^3)$, obtaining an equivalent polynomial-size LP Π'' which can be solved in polynomial time. See Ward [37, section 4.3] for more details.

We have implemented the LP Π'' on a computer, and calculated the coefficients $G_{r,1}, \ldots, G_{r,r}$ and the resulting approximation ratio for various values of r and c. The resulting approximation ratios are slightly larger than $(1 - e^{-c})/c$, but as r becomes bigger, the approximation ratio approaches $(1 - e^{-c})/c$. The function g considered in this paper is obtained by (in some sense) taking the limit $r \to \infty$.[3]

[3] In the case of maximum coverage, discussed in section 8.3, the idea of taking the limit $r \to \infty$ can be explained formally. The general form of the function g in that case is $g(A) = \sum_{x \in \mathcal{V}} \ell_{|A[x]|} w(x)$, where $\mathcal{V}$ is the universe of elements, $w \colon \mathcal{V} \to \mathbb{R}_{\geq 0}$ is the weight function, $A[x]$ is the subset of A consisting of all sets containing x, and $(\ell_t)_{t \in \mathbb{N}}$ are the coefficients determining the function g. For each rank r, one can construct an LP Π_r involving $\ell_0, \ldots, \ell_r$ that determines the best choice of coefficients. See Ward [37, section 3.4] for more details (there the sequence is known as $(\alpha_t)_{t \in \mathbb{N}}$). For $r_1 \leq r_2$, the LP Π_{r_1} is obtained from Π_{r_2} by removing all constraints involving ℓ_t for $t > r_1$. We can thus construct an infinite LP Π_∞ in infinitely many variables $(\ell_t)_{t \in \mathbb{N}}$ which extends all LPs Π_r. The program Π_∞ has the value $1 - 1/e$, attained by the unique choice of coefficients detailed in section 8.3. This leads to a limiting function g_{MC} in the case of maximum coverage. The function g considered in this paper is the unique function of the form (6.1) which reduces to g_{MC} when f is a coverage function.

6.2. Proof analysis. Another approach to understanding our choice of g is by analyzing the proof of the locality ratio of Algorithm 1, given by Theorem 5.1:

$$\frac{ce^c}{e^c-1}f(S) \geq f(O) + \sum_{i=1}^{r}[g(S) - g(S - s_i + o_i)].$$

Here $O = \{o_1, \ldots, o_r\}$ is a global optimum of f (over the matroid), $S = \{s_1, \ldots, s_r\}$, and the indices are chosen so that $S - s_i + o_i \in \mathcal{I}$ (using Brualdi's lemma). The *locality ratio* is the minimal possible value of $f(S)/f(O)$ over all locally optimal S (sets for which $g(S) \geq g(S - s_i + o_i)$ for all $i \in [r]$), in this case $(1 - e^{-c})/c$.

The analysis of Algorithm 1 relies on the fact that g satisfies (4.1) for some coefficients $\mu_A(B)$, that is

$$g_A(x) = \sum_{B \subseteq A} \mu_A(B) f_B(x).$$

We suppose that the value of g is invariant under permutations of the ground set $\mathcal{U}$. It immediately follows that $\mu_A(B)$ should depend only on $|A|, |B|$ and hence, also be invariant under permutations of $\mathcal{U}$. We obtain more constraints on $\mu_A(B)$ by examining the proof of the submodularity of g in Lemma 4.3: if $A_1 \subseteq A_2$, then

$$g_{A_2}(x) = \sum_{B \subseteq A_2} \mu_{A_2}(B) f_B(x) \overset{(*)}{\leq} \sum_{B \subseteq A_2} \mu_{A_2}(B) f_{B \cap A_1}(x) \overset{(\dagger)}{=} \sum_{B \subseteq A_1} \mu_{A_1}(B) f_B(x) = g_{A_1}(x).$$

Inequality $(*)$ requires $\mu_{A_2}(B) \geq 0$, while $(\dagger)$ implies[4] that

$$\sum_{\substack{B_2 \subseteq A_2 \\ B_2 \cap A_1 = B_1}} \mu_{A_2}(B_2) = \mu_{A_1}(B_1). \tag{6.2}$$

Summing this over all $B_1 \subseteq A_1$, we deduce

$$\sum_{B_2 \subseteq A_2} \mu_{A_2}(B_2) = \sum_{B_1 \subseteq A_1} \mu_{A_1}(B_1).$$

Without loss of generality, we can assume that the common value of this sum is 1. Since $\mu_{A_2}(B) \geq 0$, we can thus regard μ_{A_2} as a probability distribution over subsets of A. Equation (6.2) then states that if $X \sim \mu_{A_2}$, then $X \cap A_1 \sim \mu_{A_1}$.

We now show that this *restriction property* of the distributions μ_A allows us to recover μ_A up to the specific distribution P used in the definition g. Suppose that the ground set $\mathcal{U}$ were infinite, say $\mathcal{U} = \mathbb{N}$. For each $n \in \mathbb{N}$, $\mu_{[n]}$ is a distribution over subsets of $[n]$. We can define a single distribution μ over subsets of $\mathbb{N}$ by the following rule: if $X \sim \mu$, then for all $n \in \mathbb{N}$, $X \cap [n] \sim \mu_{[n]}$. Note that because each $\mu_{[n]}$ satisfies the restriction property, the distribution μ is indeed well-defined. We can think of μ as a collection of indicator random variables $(X_i)_{i \in \mathbb{N}}$ encoding the chosen subset. Although the random variables X_i are not independent, we note they are *exchangeable*—because $\mu_A(B)$ is invariant under permutations of $\mathcal{U}$, $(X_i)_{i \in \mathbb{N}}$ has the same distribution as $(X_{\pi(i)})_{i \in \mathbb{N}}$ for all permutations π on $\mathbb{N}$—and so de Finetti's

[4]In order to deduce submodularity, we in fact only need $(\dagger)$ to hold as an inequality; but for the submodularity of g to be "tight," equality is required.

theorem implies that there exists a probability distribution P supported on $[0,1]$ such that

$$\mu_A(B) = \mathop{\mathbb{E}}_{p\in P} p^{|B|}(1-p)^{|A|-|B|}.$$

As in section 4.1, we can then define the values $m_{a,b} = \mathbb{E}_{p\in P}\, p^b(1-p)^{a-b}$, and let g be given by (4.2),

$$g(A) = \sum_{B\subseteq A} m_{|A|-1,|B|-1} f(B).$$

It remains to deduce the exact distribution P. In order to do this, we consider the other property of g used in the proof of Theorem 5.1, namely, Lemma 4.1. The general form of this lemma is

$$cm_{a,b} = (a-b)m_{a-1,b} - bm_{a-1,b-1} + \begin{cases} C_0 & \text{if } b = 0, \\ 0 & \text{if } 0 < a < b, \\ C_1 & \text{if } a = b. \end{cases}$$

The locality ratio proved in Theorem 5.1 is $1/C_1$, and the proof requires $C_0 \le 0$. Lemma 4.1 is proved using integration by parts. Let $F'(p)$ be the density of P, and let $F(p)$ be an antiderivative of $F'(p)$ (not necessarily the cumulative distribution function of P). We can then restate the proof of Lemma 4.1 as follows:

$$\begin{aligned} cm_{a,b} &= c\int_0^1 F'(p)\cdot p^b(1-p)^{a-b}\,dp \\ &= cF(p)\cdot p^b(1-p)^{a-b}\big|_{p=0}^{p=1} \\ &\quad - c\int_0^1 F(p)\cdot [bp^{b-1}(1-p)^{a-b} - (a-b)p^b(1-p)^{a-b-1}]\,dp \\ &\overset{(\ddagger)}{=} [\![a=b]\!]cF(1) - [\![b=0]\!]cF(0) + (a-b)m_{a-1,b} - bm_{a-1,b-1}. \end{aligned}$$

In order for equation (‡) to hold, F must satisfy the differential equation $F' = cF$ whose solution is $F(p) \propto e^{cp}$. Since P is a probability distribution supported on $[0,1]$, we must have $F(1) - F(0) = 1$ and so $F(p) = e^{cp}/(e^c - 1)$. We deduce that $F'(p) = ce^{cp}/(e^c - 1)$, and that the locality ratio is $1/(cF(1)) = 1/F'(1) = (1-e^{-c})/c$.

7. A randomized, polynomial-time algorithm. Our analysis of Algorithm 1 supposed that we were given an oracle for computing the value of the potential function g. We now use the results of Lemma 4.5, which show that the value $g(A)$ can be approximated for any A by using a polynomial number of samples, to implement a randomized, polynomial-time approximation algorithm that does not require an oracle for g. The resulting algorithm attains the same approximation ratio as Algorithm 1 with high probability. The analysis of the modified algorithm, while somewhat tedious, is standard and in the spirit of earlier results such as Calinescu at al. [8].

The modified algorithm is shown in Algorithm 2. Algorithm 2 uses an approximation $\tilde{g}$ of g that is obtained by taking N independent random samples of f each time $\tilde{g}$ is calculated. The number of samples N depends on the parameters ϵ and α, in addition to the rank r of $\mathcal{M}$ and the size n of $\mathcal{U}$. As in Algorithm 1, ϵ governs how much an exchange must improve the current solution before it is applied, and

so affects both the approximation performance and runtime of the algorithm. The additional parameter α controls the probability that Algorithm 2 fails to produce a $(\frac{1-e^{-c}}{c} - \epsilon)$-approximate solution. Specifically, we show that Algorithm 2 fails with probability at most $O(n^{-\alpha})$.

For the analysis, we assume that $\epsilon \leq 1$ and $r \geq 2$, which imply that $\epsilon_2 \leq 1/12$.

ALGORITHM 2. THE NON-OBLIVIOUS LOCAL SEARCH ALGORITHM.

Input: $\mathcal{M} = (\mathcal{U}, \mathcal{I})$, f, c, ϵ, α
Set $\epsilon_2 = \frac{\epsilon}{4rH_r}$
Set $I = \left(\left(\frac{1+\epsilon_2}{1-\epsilon_2}\right)(2 + 3r\epsilon_2) - 1\right)\epsilon_2^{-1}$
Set $N = \frac{1}{2}\left(\frac{ce^c}{e^c-1} \cdot \frac{H_n}{\epsilon_2}\right)^2 \ln\left((I+1)rn^{1+\alpha}\right)$
Let $\tilde{g}$ be an approximation to g computed by taking N random samples
Let S_{init} be the result of running the standard greedy algorithm on $(\mathcal{M}, \tilde{g})$
$S \leftarrow S_{\text{init}}$
$v \leftarrow \tilde{g}(S_{\text{init}})$
for $i \leftarrow 1$ **to** I **do** {`Search for at most` I `iterations`}
  done $\leftarrow$ **true**
  foreach *element* $x \in S$ *and* $y \in \mathcal{U} \setminus S$ **do**
    $S' \leftarrow S - x + y$
    if $S' \in \mathcal{I}$ **then**
      $v' \leftarrow \tilde{g}(S')$
      if $v' > (1+\epsilon_2)v$ **then** {`An improved solution` S' `was found`}
        $v \leftarrow v'$ and $S \leftarrow S'$ {`update` v `and` S}
        done $\leftarrow$ **false**
        break {`and continue to the next iteration`}
  if done **then return** S {`No improvement was found, return local optimum`}
return `Error (Search did not converge in` I `iterations)`

The local search routine in Algorithm 2 runs some number I of iterations, signaling an error if it fails to converge to a local optimum after this many improvements. In each iteration, the algorithm searches through all possible solutions $S' = S - x + y$, sampling the value $\tilde{g}(S')$ if $S' \in \mathcal{I}$. If the sampled value of $\tilde{g}(S')$ exceeds the sampled value for $\tilde{g}(S)$ by a factor of at least $(1 + \epsilon_2)$, the algorithm updates S and moves to the next iteration. Otherwise, it returns the current solution. Note that we store the last sampled value $\tilde{g}(S)$ of the current solution in v, rather than resampling $\tilde{g}(S)$ each time we check an improvement S'.

The analysis of Algorithm 2 follows the same general pattern as that presented in the previous section. Here however, we must address the fact that $\tilde{g}$ does not always agree with g. First, we estimate the probability that all of the computations of $\tilde{g}$ made by Algorithm 2 are reasonably close to the value of g.

LEMMA 7.1. *With probability* $1 - O(n^{-\alpha})$, *we have* $|\tilde{g}(A) - g(A)| \leq \epsilon_2 g(A)$ *for all sets* A *for which Algorithm* 2 *computes* $\tilde{g}(A)$.

Proof. We first bound the total number of sets A for which Algorithm 2 computes $\tilde{g}(A)$. The initial greedy phase requires fewer than rn evaluations, as does each of the I iterations of the local phase. The total number of evaluations is therefore less than $(I+1)rn$.

Algorithm 2 uses

$$N = \frac{1}{2}\left(\frac{ce^c}{e^c-1}\cdot\frac{H_n}{\epsilon_2}\right)^2 \ln\left((I+1)rn^{1+\alpha}\right)$$

samples for every computation of $\tilde{g}(A)$. By Lemma 4.5, the probability that we have $|g(A)-\tilde{g}(A)| \geq \epsilon_2 g(A)$ for any given set A is then $\delta = O(\frac{1}{(I+1)rn^{1+\alpha}})$.

Let the sets at which Algorithm 2 evaluates $\tilde{g}$ be $A_1,\ldots,A_M$, where $M \leq (I+1)rn$; both M and the sets A_i are random variables depending on the execution of the algorithm. The events $|g(A_i)-\tilde{g}(A_i)| \geq \epsilon_2 g(A_i)$ are independent, and so from the union bound, the probability that at least one of the sets A_i does not satisfy the desired error bound is at most $O(\frac{(I+1)rn}{(I+1)rn^{1+\alpha}}) = O(n^{-\alpha})$. □

We call the condition that $|g(A)-\tilde{g}(A)| \leq \epsilon_2 g(A)$ for all sets A considered by Algorithm 2 the *sampling assumption*. Lemma 7.1 shows that the sampling assumption holds with high probability.

Now, we must adapt the analysis of section 5, which holds when g is computed exactly, to the setting in which g is computed approximately. In Theorem 5.5, we showed that $g(S_{\text{init}})$ is within a constant factor of the largest possible value that g could take on any set $A \subseteq \mathcal{U}$. Then, because the algorithm always improved g by a factor of at least $(1+\epsilon_1)$, we could bound the number of local search iterations that it performed. Finally, we applied Theorem 5.1 to translate the local optimality of S with respect to g into a lower bound on $f(S)$.

Here we follow the same general approach. First, we derive the following result, which shows that the initial value $\tilde{g}(S_{\text{init}})$ is within a constant factor of the maximum value $\tilde{g}^*$ of $\tilde{g}(A)$ on any set A considered by Algorithm 2.[5]

LEMMA 7.2. *Suppose that the sampling assumption is true, and let $\tilde{g}^*$ be the maximum value of $\tilde{g}(A)$ over all sets A considered by Algorithm* 2. *Then,*

$$(2+3r\epsilon_2)\left(\frac{1+\epsilon_2}{1-\epsilon_2}\right)\tilde{g}(S_{\text{init}}) \geq \tilde{g}^*.$$

Proof. The standard greedy algorithm successively chooses a sequence of sets $\emptyset = S_0, S_1, \ldots, S_r = S_{\text{init}}$, where each S_i for $i > 0$ satisfies $S_i = S_{i-1} + s_i$ for some element $s_i \in \mathcal{U} \setminus S_{i-1}$. The element s_i is chosen at each phase according to the formula

$$s_i = \operatorname*{argmax}_{\substack{x \in \mathcal{U}\setminus S_{i-1} \\ \text{s.t. } S_{i-1}+x \in \mathcal{I}}} \tilde{g}(S_{i-1}+x).$$

Let O be any base of $\mathcal{M}$ on which g attains its maximum value. According to Brualdi's lemma, we can index $O = \{o_1,\ldots,o_r\}$ so that $o_i = \pi(s_i)$ for all $i \in [r]$. Then, the set $S_{i-1}+o_i$ is independent for all $i \in [r]$. Thus, we must have

$$\tilde{g}(S_{i-1}+s_i) \geq \tilde{g}(S_{i-1}+o_i)$$

for all $i \in [r]$. In order to use monotonicity and submodularity, we translate this into an inequality for g. From the sampling assumption, we have

$$(1+\epsilon_2)g(S_{i-1}+s_i) \geq \tilde{g}(S_{i-1}+s_i) \geq \tilde{g}(S_{i-1}+o_i) \geq (1-\epsilon_2)g(S_{i-1}+o_i).$$

[5] A similar result for the greedy algorithm applied to an approximately calculated submodular function is given by Calinescu et al. [8]. However, in their model, the *marginals* of a submodular function are approximately calculated, while in ours, the *value* of the submodular function is approximately calculated. For the sake of completeness, we provide a complete proof for our setting.

Then, since $(1+\epsilon_2)/(1-\epsilon_2) \leq 1+3\epsilon_2$ for all $\epsilon_2 \leq 1/3$,

$$(1+3\epsilon_2)g(S_{i-1}+s_i) \geq \frac{(1+\epsilon_2)}{(1-\epsilon_2)}g(S_{i-1}+s_i) \geq g(S_{i-1}+o_i).$$

Subtracting $g(S_{i-1})$ from each side above, we obtain

$$3\epsilon_2 g(S_i) + g_{S_{i-1}}(s_i) \geq g_{S_{i-1}}(o_i)$$

for each $i \in [r]$. Summing the resulting r inequalities, we obtain a telescoping summation, which gives

$$\begin{aligned} 3\epsilon_2 \sum_{i=1}^{r} g(S_i) + g(S_{\text{init}}) &\geq \sum_{i=1}^{r} g_{S_{i-1}}(o_i) \geq \sum_{i=1}^{r} g_{S_{\text{init}}}(o_i) \\ &\geq g_{S_{\text{init}}}(O) = g(O \cup S_{\text{init}}) - g(S_{\text{init}}), \end{aligned}$$

where we have used submodularity of g for the second and third inequalities. Then, using the monotonicity of g, we have $3\epsilon_2 \sum_{i=1}^{r} g(S_{\text{init}}) \geq 3\epsilon_2 \sum_{i=1}^{r} g(S_i)$ on the left, and $g(O \cup S_{\text{init}}) \geq g(S_{\text{init}})$ on the right, and so

$$3r\epsilon_2 g(S_{\text{init}}) + 2g(S_{\text{init}}) \geq g(O). \tag{7.1}$$

Finally, by the sampling assumption we must have $\tilde{g}(S_{\text{init}}) \geq (1-\epsilon_2)g(S_{\text{init}})$ and also $(1+\epsilon_2)g(O) \geq (1+\epsilon_2)g(A) \geq \tilde{g}(A)$ for any set A considered by the algorithm. Thus, (7.1) implies

$$(2+3r\epsilon_2)\left(\frac{1+\epsilon_2}{1-\epsilon_2}\right)\tilde{g}(S_{\text{init}}) \geq \tilde{g}^*. \qquad \square$$

The next difficulty we must overcome is that the final set S produced by Algorithm 2 is (approximately) locally optimal only with respect to the sampled function $\tilde{g}(S)$. In order to use Theorem 5.1 to obtain a lower bound on $f(S)$, we must show that S is approximately locally optimal with respect to g as well. We accomplish this in our next lemma, by showing that any significant improvement in $\tilde{g}$ must correspond to a (somewhat less) significant improvement in g.

LEMMA 7.3. *Suppose that the sampling assumption holds and that* $\tilde{g}(A) \leq (1+\epsilon_2)\tilde{g}(B)$ *for some pair of sets* A, B *considered by Algorithm* 2. *Then*

$$g(A) \leq (1+4\epsilon_2)g(B).$$

Proof. From the sampling assumption, we have

$$(1-\epsilon_2)g(A) \leq \tilde{g}(A) \leq (1+\epsilon_2)\tilde{g}(B) \leq (1+\epsilon_2)(1+\epsilon_2)g(B).$$

Thus

$$g(A) \leq \frac{(1+\epsilon_2)^2}{1-\epsilon_2}g(B) \leq (1+4\epsilon_2)g(B),$$

where the second inequality holds since $\epsilon_2 \leq 1/5$. $\square$

We now prove our main result.

THEOREM 7.4. *Algorithm* 2 *runs in time* $\tilde{O}(r^4 n\epsilon^{-3}\alpha)$ *and returns a* $(\frac{1-e^{-c}}{c} - \epsilon)$*-approximation with probability* $1 - O(n^{-\alpha})$.

Proof. As in the proof of Theorem 5.5, we consider some arbitrary instance $(\mathcal{M} = (\mathcal{U}, \mathcal{I}), f)$ of monotone submodular matroid maximization with upper bound c on the curvature of f, and let O be an optimal solution of this instance. We shall show that if the sampling assumption holds, Algorithm 2 returns a solution S satisfying $f(S) \geq (\frac{1-e^{-c}}{c} - \epsilon)f(O)$. Lemma 7.1 shows that this happens with probability $1 - O(n^{-\alpha})$.

As in Algorithm 2, set

$$I = \left(\left(\frac{1+\epsilon_2}{1-\epsilon_2}\right)(2+3r\epsilon_2) - 1\right)\epsilon_2^{-1}.$$

Suppose that the sampling assumption holds, and let g^* be the maximum value taken by $\tilde{g}(A)$ for any set A considered by Algorithm 2. At each iteration of Algorithm 2, either a set S is returned, or the value v is increased by a factor of at least $(1+\epsilon_2)$. Suppose that the local search phase of Algorithm 2 fails to converge to a local optimum after I steps, and so does not return a solution S. Then we must have

$$v \geq (1+\epsilon_2)^I \tilde{g}(S_{\text{init}}) > (1+I\epsilon_2)\tilde{g}(S_{\text{init}}) = \left(\frac{1+\epsilon_2}{1-\epsilon_2}\right)(2+3r\epsilon_2)\tilde{g}(S_{\text{init}}) \geq g^*,$$

where the last inequality follows from Lemma 7.2. But, then we must have $\tilde{g}(A) > g^*$ for some set A considered by the algorithm. Thus Algorithm 2 must produce a solution S.

As in Theorem 5.5, we apply Theorem 5.1 to the bases S and O, indexing S and O as in the theorem so that $S - s_i + o_i \in \mathcal{I}$ for all $i \in [r]$, to obtain

$$\frac{ce^c}{e^c-1}f(S) \geq f(O) + \sum_{i=1}^{r}[g(S) - g(S - s_i + o_i)]. \tag{7.2}$$

Then, since Algorithm 2 returned S, we must then have

$$\tilde{g}(S - s_i + o_i) \leq (1+\epsilon_2)\tilde{g}(S)$$

for all $i \in [r]$. From Lemma 7.3 we then have

$$g(S - s_i + o_i) \leq (1+4\epsilon_2)g(S)$$

for all $i \in [r]$. Summing the resulting r inequalities gives

$$\sum_{i=1}^{r}[g(S) - g(S - s_i + o_i)] \geq -4r\epsilon_2 g(S).$$

Applying Theorem 5.5 and the upper bound on $g(S)$ from Lemma 4.4 in (7.2), we then have

$$\begin{aligned}\frac{ce^c}{e^c-1}f(S) &\geq f(O) - 4r\epsilon_2 g(S) \geq f(O) - \frac{ce^c}{e^c-1}4r\epsilon_2 H_r f(S)\\ &\geq f(O) - \frac{ce^c}{e^c-1}4r\epsilon_2 H_r f(O).\end{aligned}$$

Rewriting this inequality using the definition $\epsilon_2 = \frac{\epsilon}{4rH_r}$ then gives

$$f(S) \geq \left(\frac{1-e^{-c}}{c} - \epsilon\right)f(O).$$

The running time of Algorithm 2 is dominated by the number of calls it makes to the value oracle for f. We note, as in the proof of Lemma 7.1, that the algorithm evaluates $\tilde{g}(A)$ on $O(rnI)$ sets A. Each evaluation requires N samples of f, and so the resulting algorithm requires

$$O(rnIN) = \tilde{O}(rn\epsilon_2^{-3}\alpha) = \tilde{O}(r^4 n\epsilon^{-3}\alpha)$$

calls to the value oracle for f. ∎

8. Extensions. The algorithm presented in section 3 produces a $(\frac{1-e^{-c}}{c} - \epsilon)$-approximation for any $\epsilon > 0$, and it requires knowledge of c. In this section we show how to produce a clean $(1 - e^{-c})/c$-approximation, and how to dispense with the knowledge of c. Unfortunately, we are unable to combine both improvements for technical reasons.

It will be useful to define the function

$$\rho(c) = \frac{1 - e^{-c}}{c},$$

which gives the optimal approximation ratio.

8.1. Clean approximation. In this section, we assume c is known, and our goal is to obtain a $\rho(c)$ approximation algorithm. We accomplish this by combining the algorithm from section 7 with partial enumeration.

For $x \in \mathcal{U}$ we consider the contracted matroid $\mathcal{M}/x$ on $\mathcal{U} - x$ whose independent sets are given by $\mathcal{I}_x = \{A \subseteq \mathcal{U} - x : A + x \in \mathcal{I}\}$, and the contracted submodular function (f/x) which is given by $(f/x)(A) = f(A + x)$. It is easy to show that this function is a monotone submodular function whenever f is, and has curvature at most that of f. Then, for each $x \in \mathcal{U}$, we apply Algorithm 2 to the instance $(\mathcal{M}/x, f/x)$ to obtain a solution S_x. We then return $S_x + x$ for the element $x \in \mathcal{U}$ maximizing $f(S_x + x)$.

Nemhauser, Wolsey, and Fisher [31] analyze this technique in the case of submodular maximization over a uniform matroid, and Khuller, Moss, and Naor [27] make use of the same technique in the restricted setting of budgeted maximum coverage. Calinescu et al. [7] use a similar technique to eliminate the error term from the approximation ratio of the continuous greedy algorithm for general monotone submodular matroid maximization. Our proof relies on the following general claim.

LEMMA 8.1. *Suppose $A \subseteq O$ and $B \subseteq \mathcal{U} \setminus A$ satisfy $f(A) \geq (1 - \theta_A)f(O)$ and $f_A(B) \geq (1 - \theta_B)f_A(O \setminus A)$. Then*

$$f(A \cup B) \geq (1 - \theta_A\theta_B)f(O).$$

Proof. We have

$$\begin{aligned} f(A \cup B) &= f_A(B) + f(A) \\ &\geq (1 - \theta_B)f_A(O \setminus A) + f(A) \\ &= (1 - \theta_B)f(O) + \theta_B f(A) \\ &\geq (1 - \theta_B)f(O) + \theta_B(1 - \theta_A)f(O) \\ &= (1 - \theta_A\theta_B)f(O). \end{aligned}$$ ∎

Using Lemma 8.1 we show that the partial enumeration procedure gives a clean $\rho(c)$-approximation algorithm.

THEOREM 8.2. *The partial enumeration algorithm runs in time $\tilde{O}(r^7 n^2 \alpha)$, and with probability $1 - O(n^{-\alpha})$, the algorithm has an approximation ratio of $\rho(c)$.*

Proof. Let $O = \{o_1, \ldots, o_r\}$ be an optimal solution to some instance $(\mathcal{M}, f)$. Since the submodularity of f implies

$$\sum_{i=1}^{r} f(o_i) \geq f(O),$$

there is some $x \in O$ such that $f(x) \geq f(O)/r$. Take $A = \{x\}$ and $B = S_x$ in Lemma 8.1. Then, from Theorem 7.4 we have $(f/x)(S_x) \geq (\rho(c) - \epsilon)f(O)$ with probability $1 - O(n^{-\alpha})$ for any ϵ. We set $\epsilon = (1 - \rho(c))/r$. Then, substituting $\theta_A = 1 - 1/r$ and $\theta_B = 1 - \rho(c) + (1 - \rho(c))/r$, we deduce that the resulting approximation ratio in this case is

$$\begin{aligned} 1 - \left(1 - \frac{1}{r}\right)\left(1 - \rho(c) + \frac{1 - \rho(c)}{r}\right) &= 1 - \left(1 - \frac{1}{r}\right)\left(1 + \frac{1}{r}\right)(1 - \rho(c)) \\ &\geq 1 - (1 - \rho(c)) = \rho(c). \end{aligned}$$

The partial enumeration algorithm simply runs Algorithm 2 n times, using $\epsilon = O(r^{-1})$ and so its running time is $\tilde{O}(r^7 n^2 \alpha)$. ∎

8.2. Unknown curvature. In this section, we remove the assumption that c is known, but retain the error parameter ϵ. The key observation is that if a function has curvature c, then it also has curvature c' for any $c' \geq c$. This, combined with the continuity of ρ, allows us to "guess" an approximate value of c.

Given ϵ, consider the following algorithm. Define the set C of curvature approximations by

$$C = \{k\epsilon : 1 \leq k \leq \lfloor \epsilon^{-1} \rfloor\} \cup \{1\}.$$

For each guess $c' \in C$, we run the main algorithm with that setting of c' and error parameter $\epsilon/2$ to obtain a solution $S_{c'}$. Finally, we output the set $S_{c'}$ maximizing $f(S_{c'})$.

THEOREM 8.3. *Suppose f has curvature c. The unknown curvature algorithm runs in time $\tilde{O}(r^4 n \epsilon^{-4} \alpha)$, and with probability $1 - O(n^{-\alpha})$, the algorithm has an approximation ratio of $\rho(c) - \epsilon$.*

Proof. From the definition of C it is clear that there is some $c' \in C$ satisfying $c \leq c' \leq c + \epsilon$. Since f has curvature c, the set $S_{c'}$ is a $(\rho(c') - \epsilon/2)$-approximation. Elementary calculus shows that on $(0, 1]$, the derivative of ρ is at least $-1/2$, and so we have

$$\rho(c') - \epsilon/2 \geq \rho(c + \epsilon) - \epsilon/2 \geq \rho(c) - \epsilon/2 - \epsilon/2 = \rho(c) - \epsilon. \qquad \square$$

8.3. Maximum coverage. In the special case that f is given explicitly as a *coverage function*, we can evaluate the potential function g exactly in polynomial time. A (weighted) coverage function is a particular kind of monotone submodular function that may be given in the following way. There is a universe $\mathcal{V}$ with nonnegative weight function $w \colon \mathcal{V} \to \mathbb{R}_{\geq 0}$. The weight function is extended to subsets of $\mathcal{V}$ linearly, by letting $w(S) = \sum_{s \in S} w(s)$ for all $S \subseteq \mathcal{V}$. Additionally, we are given a family $\{V_a\}_{a \in \mathcal{U}}$ of subsets of $\mathcal{V}$, indexed by a set $\mathcal{U}$. The function f is then defined over the index set $\mathcal{U}$, and $f(A)$ is simply the total weight of all elements of $\mathcal{V}$ that are covered by those sets whose indices appear in A. That is, $f(A) = w\left(\bigcup_{a \in A} V_a\right)$.

We now show how to compute the potential function g exactly in this case. For a set $A \subseteq \mathcal{U}$ and an element $x \in \mathcal{V}$, we denote by $A[x]$ the collection $\{a \in A : x \in V_a\}$

of indices a such that x is in the set V_a. Then, recalling the definition of $g(A)$ given in (4.2), we have

$$\begin{aligned} g(A) &= \sum_{B \subseteq A} m_{|A|-1,|B|-1} f(B) \\ &= \sum_{B \subseteq A} m_{|A|-1,|B|-1} \sum_{x \in \bigcup_{b \in B} V_b} w(x) \\ &= \sum_{x \in \mathcal{V}} w(x) \sum_{\substack{B \subseteq A \ s.t. \\ A[x] \cap B \neq \emptyset}} m_{|A|-1,|B|-1}. \end{aligned}$$

Consider the coefficient of $w(x)$ in the above expression for $g(A)$. We have

$$\begin{aligned} \sum_{\substack{B \subseteq A \ s.t. \\ A[x] \cap B \neq \emptyset}} m_{|A|-1,|B|-1} &= \sum_{B \subseteq A} m_{|A|-1,|B|-1} - \sum_{B \subseteq A \setminus A[x]} m_{|A|-1,|B|-1} \\ &= \sum_{i=0}^{|A|} \binom{|A|}{i} m_{|A|-1,i-1} - \sum_{i=0}^{|A \setminus A[x]|} \binom{|A \setminus A[x]|}{i} m_{|A|-1,i-1} \\ &= \sum_{i=0}^{|A|} \binom{|A|}{i} \mathop{\mathbb{E}}_{p \sim P}[p^{i-1}(1-p)^{|A|-i}] \\ &\quad - \sum_{i=0}^{|A \setminus A[x]|} \binom{|A \setminus A[x]|}{i} \mathop{\mathbb{E}}_{p \sim P}[p^{i-1}(1-p)^{|A|-i}] \\ &= \mathop{\mathbb{E}}_{p \sim P}\left[\frac{1}{p} \sum_{i=0}^{|A|} \binom{|A|}{i} p^i (1-p)^{|A|-i} \right. \\ &\qquad \left. - \frac{(1-p)^{|A[x]|}}{p} \sum_{i=1}^{|A \setminus A[x]|} \binom{|A \setminus A[x]|}{i} p^i (1-p)^{|A \setminus A[x]|-i} \right] \\ &= \mathop{\mathbb{E}}_{p \sim P}\left[\frac{1-(1-p)^{|A[x]|}}{p} \right]. \end{aligned}$$

Thus, if we define

$$\ell_k = \mathop{\mathbb{E}}_{p \sim P}\left[\frac{1-(1-p)^k}{p} \right]$$

we have

$$g(A) = \sum_{x \in \mathcal{V}} \ell_{|A[x]|} w(x),$$

and so to compute g, it is sufficient to maintain for each element $x \in \mathcal{V}$ a count of the number of sets $A[x]$ with indices in A that contain x. Using this approach, each change in $g(S)$ resulting from adding an element x to S and removing an element e from S during one step of the local search phase of Algorithm 1 can be computed in time $O(|\mathcal{V}|)$.

We further note that the coefficients ℓ_k are easily calculated using the following recurrence. For $k = 0$,

$$\ell_0 = \mathop{\mathbb{E}}_{p \sim P}\left[\frac{1-(1-p)^0}{p} \right] = 0,$$

while for $k > 0$,

$$\ell_{k+1} = \mathop{\mathbb{E}}_{p\sim P}\left[\frac{1-(1-p)^{k+1}}{p}\right] = \mathop{\mathbb{E}}_{p\sim P}\left[\frac{1-(1-p)^k + p(1-p)^k}{p}\right]$$
$$= \ell_k + \mathop{\mathbb{E}}_{p\sim P}(1-p)^k = \ell_k + m_{k,0}.$$

The coefficients ℓ_k obtained in this fashion in fact correspond (up to a constant scaling factor) to those used to define the non-oblivious coverage potential in [17], showing that our algorithm for monotone submodular matroid maximization is indeed a generalization of the algorithm already obtained in the coverage case.

When all subsets V_a consist of the same element $V_a = \{x\}$ of weight $w(x) = 1$,

$$g(A) = \sum_{B\subseteq A} m_{|A|-1,|B|-1} f(B) = \sum_{\substack{B\subseteq A \ s.t. \\ B\neq\emptyset}} m_{|A|-1,|B|-1} = \tau(A).$$

(The quantity $\tau(A)$ is defined in section 4.1.) On the other hand, $g(A) = \ell_{|A|}$. We conclude that $\tau(A) = \ell_{|A|}$.

REFERENCES

[1] A. A. Ageev and M. I. Sviridenko, *Pipage rounding: A new method of constructing algorithms with proven performance guarantee*, J. Comb. Optim., 8 (2004), pp. 307–328.

[2] P. Alimonti, *New local search approximation techniques for maximum generalized satisfiability problems*, in CIAC: Proceedings of the 2nd Italian Conference on Algorithms and Complexity, Springer-Verlag, New York, 1994, pp. 40–53.

[3] B. V. Ashwinkumar and J. Vondrák, *Fast algorithms for maximizing submodular functions*, in Proceedings of the 25th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA), SIAM, Philadelphia, 2014, pp. 1497–1514.

[4] R. A. Brualdi, *Comments on bases in dependence structures*, Bull. Austral. Math. Soc., 1 (1969), pp. 161–167.

[5] N. Buchbinder, M. Feldman, J. (Seffi) Naor, and R. Schwartz, *A tight linear time* (1/2)*-approximation for unconstrained submodular maximization*, in FOCS, IEEE, Piscataway, NJ, 2012, pp. 649–658.

[6] N. Buchbinder, M. Feldman, J. (Seffi) Naor, and R. Schwartz, *Submodular maximization with cardinality constraints*, in Proceedings of the 25th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA), SIAM, Philadelphia, 2014, pp. 1433–1452.

[7] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, *Maximizing a submodular set function subject to a matroid constraint (Extended abstract)*, in 12th International IPCO Conference on Integer Programming and Combinatorial Optimization, Springer, Berlin, 2007, pp. 182–196.

[8] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, *Maximizing a monotone submodular function subject to a matroid constraint*, SIAM J. Comput., 40 (2011), pp. 1740–1766.

[9] C. Checkuri, J. Vondrák, and R. Zenklusen, *Submodular function maximization via the multilinear relaxation and contention resolution schemes*, STOC, San Jose, CA, 2011.

[10] C. Chekuri, J. Vondrak, and R. Zenklusen, *Dependent randomized rounding via exchange properties of combinatorial structures*, in Proceedings of the 2010 IEEE 51st Annual Symposium on FOCS, IEEE, Piscataway, NJ, 2010, pp. 575–584.

[11] J. Edmonds, *Matroids and the greedy algorithm*, Math. Program., 1 (1971), pp. 127–136.

[12] U. Feige, *A threshold of ln n for approximating set cover*, J. ACM, 45 (1998), pp. 634–652.

[13] U. Feige, V. S. Mirrokni, and J. Vondrak, *Maximizing non-monotone submodular functions*, in FOCS, IEEE, Piscataway, NJ, 2007, pp. 461–471.

[14] M. Feldman, J. Naor, and R. Schwartz, *A unified continuous greedy algorithm for submodular maximization*, in FOCS, IEEE, Piscataway, NJ, 2011, pp. 570–579.

[15] M. Feldman, J. (Seffi) Naor, and R. Schwartz, *Nonmonotone submodular maximization via a structural continuous greedy algorithm*, in ICALP, Springer-Verlag, New York, 2011, pp. 342–353.

[16] M. Feldman, J. (Seffi) Naor, R. Schwartz, and J. Ward, *Improved approximations for k-exchange systems*, in 19th Annual European Symposium on Algorithms, Saarbrücken, Germany, Springer, New York, 2011, pp. 784–798.
[17] Y. Filmus and J. Ward, *The power of local search: Maximum coverage over a matroid*, in STACS, Leibniz-Zentrum fur Informatik GmbH, Schloss Dagstuhl, Wadern, Germany, 2012, pp. 601–612.
[18] Y. Filmus and J. Ward, *A tight combinatorial algorithm for submodular maximization subject to a matroid constraint*, in FOCS, IEEE, Piscataway, NJ, 2012, pp. 659–668.
[19] Y. Filmus and J. Ward, *A Tight Combinatorial Algorithm for Submodular Maximization Subject to a Matroid Constraint*, preprint, arXiv:1204.4526, 2012.
[20] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, *An analysis of approximations for maximizing submodular set functions*—II, in Polyhedral Combinatorics, Springer, Berlin, 1978, pp. 73–87.
[21] M. Frank and P. Wolfe, *An algorithm for quadratic programming*, Naval Res. Logist., 3 (1956), pp. 95–110.
[22] S. Oveis Gharan and J. Vondrák, *Submodular maximization by simulated annealing*, in SODA, SIAM, Philadelphia, 2011, pp. 1098–1116.
[23] M. X. Goemans and J. Kleinberg, *An improved approximation ratio for the minimum latency problem*, Math. Program., 82 (1998), pp. 111–124.
[24] P. R. Goundan and A. S. Schulz, *Revisiting the Greedy Approach to Submodular Set Function Maximization*, manuscript, 2007.
[25] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani, *Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP*, J. ACM, 50 (2003), pp. 795–824.
[26] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani, *On syntactic versus computational views of approximability*, SIAM J. Comput., 28 (1999), pp. 164–191.
[27] S. Khuller, A. Moss, and J. (Seffi) Naor, *The budgeted maximum coverage problem*, Inform. Process. Lett., 70 (1999), pp. 39–45.
[28] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko, *Non-monotone submodular maximization under matroid and knapsack constraints*, in STOC, ACM, New York, 2009, pp. 323–332.
[29] J. Lee, M. Sviridenko, and J. Vondrák, *Submodular maximization over multiple matroids via generalized exchange properties*, Math. Oper. Res., 35 (2010), pp. 795–806.
[30] G. L. Nemhauser and L. A. Wolsey, *Best algorithms for approximating the maximum of a submodular set function*, Math. Oper. Res., 3 (1978), pp. 177–188.
[31] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, *An analysis of approximations for maximizing submodular set functions*—I, Math. Program., 14 (1978), pp. 265–294.
[32] R. Rado, *Note on independence functions*, Proc. London Math. Soc. (3), 7 (1957), pp. 300–320.
[33] M. Sviridenko and J. Ward, *Tight Bounds for Submodular and Supermodular Optimization with Bounded Curvature*, preprint, arXiv:1311.4728, 2013.
[34] J. Vondrák, *Optimal approximation for the submodular welfare problem in the value oracle model*, in STOC, ACM, New York, 2008, pp. 67–74.
[35] J. Vondrák, *Submodularity and curvature: The optimal algorithm*, in RIMS Kokyuroku Bessatsu B23, S. Iwata, ed., Kyoto University, Kyoto, 2010, pp. 253–266.
[36] J. Ward, *A* $(k+3)/2$*-approximation algorithm for monotone submodular* k*-set packing and general* k*-exchange systems*, in STACS, Leibniz-Zentrum fur Informatik GmbH, Schloss Dagstuhl, Wadern, Germany, 2012, pp. 42–53.
[37] J. Ward, *Oblivious and Non-Oblivious Local Search for Combinatorial Optimization*, Ph.D. thesis, University of Toronto, Toronto, 2012.