Randomized Composable Core-sets for Distributed Submodular Maximization

Vahab Mirrokni
Google Research
New York, NY
mirrokni@google.com

Morteza Zadimoghaddam
Google Research
New York, NY
zadim@google.com

ABSTRACT

An effective technique for solving optimization problems over massive data sets is to partition the data into smaller pieces, solve the problem on each piece and compute a representative solution from it, and finally obtain a solution inside the union of the representative solutions for all pieces. This technique can be captured via the concept of *composable core-sets*, and has been recently applied to solve diversity maximization problems as well as several clustering problems [7, 15, 8]. However, for coverage and submodular maximization problems, impossibility bounds are known for this technique [15]. In this paper, we focus on efficient construction of a randomized variant of composable core-sets where the above idea is applied on a *random clustering* of the data. We employ this technique for the coverage, monotone and non-monotone submodular maximization problems. Our results significantly improve upon the hardness results for non-randomized core-sets, and imply improved results for submodular maximization in a distributed and streaming settings. The effectiveness of this technique has been confirmed empirically for several machine learning applications [22], and our proof provides a theoretical foundation to this idea.

In summary, we show that a simple greedy algorithm results in a $1/3$ -approximate randomized composable core-set for submodular maximization under a cardinality constraint. Our result also extends to non-monotone submodular functions, and leads to the first 2-round MapReduce-based constant-factor approximation algorithm with $O(n)$ total communication complexity for either monotone or non-monotone functions. Finally, using an improved analysis technique and a new algorithm *PseudoGreedy*, we present an improved 0.545 -approximation algorithm for monotone submodular maximization, which is in turn the first MapReduce-based algorithm beating factor $1/2$ in a constant number of rounds.

Categories and Subject Descriptors

F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems; G.2.2 [Discrete Mathematics]: Combinatorics—*Combinatorial algorithms*

Keywords

Submodular Maximization; Distributed algorithms; Streaming Algorithms, Core-sets; Randomized Composable Core-sets; MapReduce Algorithms

1. INTRODUCTION

An effective way of processing massive data is to first extract a compact representation of the data and then perform further processing only on the representation itself. This approach significantly reduces the cost of processing, communicating and storing the data, as the representation size can be much smaller than the size of the original data set. Typically, the representation provides a smooth tradeoff between its size and the representation accuracy. Examples of this approach include techniques such as sampling, sketching, (composable) core-sets and mergeable summaries. Among these techniques, the concept of composable core-sets has been employed in several distributed optimization models such as nearest neighbor search [1], and the streaming and MapReduce models [22, 7, 15, 5, 8]. Roughly speaking, the main idea behind this technique is as follows: First partition the data into smaller parts. Then compute a representative solution, referred to as a *core-set*, from each part. Finally, obtain a solution by solving the optimization problem over the union of core-sets for all parts. While this technique has been successfully applied to diversity maximization and clustering problems [7, 15, 8], for coverage and submodular maximization problems, impossibility bounds are known for this technique [15].

In this paper, we focus on efficient construction of a randomized variant of composable core-sets where the above idea is applied on a *random clustering* of the data. We employ this technique for the coverage, monotone and non-monotone submodular problems. Our results significantly improve upon the hardness results for non-randomized core-sets, and imply improved results for submodular maximization in a distributed and streaming settings. The effectiveness of this technique has been confirmed empirically for several machine learning applications [22], and our proof provides a theoretical foundation to this idea. Let us first define this concept, and then discuss its applications, and our results.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s). Copyright is held by the author/owner(s).

STOC'15, June 14–17, 2015, Portland, Oregon, USA.

ACM 978-1-4503-3536-2/15/06.

<http://dx.doi.org/10.1145/2746539.2746624>.

1.1 Preliminaries

Here, we discuss the formal problem definition, and the distributed model motivating it.

Submodular Functions. We start by defining submodular functions¹. Let $\mathbb{N}$ be a ground set of items with cardinality $n = |\mathbb{N}|$. Consider a set function $f : 2^{\mathbb{N}} \rightarrow \mathbb{R}^+ \cup \{0\}$. We say function f is monotone if for any two subsets $X \subseteq Y \subseteq \mathbb{N}$, $f(X) \leq f(Y)$. We say function f is submodular if and only if for any two subsets $X \subseteq Y \subseteq \mathbb{N}$, and an item $x \in \mathbb{N} \setminus Y$, we have the property of diminishing returns, i.e.,

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y).$$

Given an integer size constraint k , we let f_k be

$$f_k(S) \stackrel{\text{def}}{=} \max_{S' \subseteq S, |S'| \leq k} f(S').$$

The submodular maximization problem with a cardinality constraint is as follows: given a parameter k and a value oracle access to a non-negative submodular function $f : 2^{\mathbb{N}} \rightarrow \mathbb{R}^+ \cup \{0\}$, find a subset S of cardinality at most k with the maximum value $f(S)$. The most common algorithm for solving the above problem is algorithm *Greedy* which is as follows: start from an empty set $S = \emptyset$, and in k iterations, find an item x with maximum marginal f value for S (i.e., $x = \arg \max_{y \in \mathbb{N}} f(S \cup \{y\}) - f(S)$) and add this item x to S . We refer to this algorithm as algorithm *Greedy* and note that it is a $(1 - \frac{1}{e})$ -approximation for monotone submodular maximization problem with a cardinality constraint.

Randomized Composable Core-sets. In this paper, we assume that all n items of $\mathbb{N}$ do not fit on one machine, and we need to apply a distributed algorithm to solve submodular maximization problem. To deal with this issue, we consider distributing items of $\mathbb{N}$ into m machines with indices $\{1, \dots, m\}$, where each item goes to C randomly chosen machines. Let $\{T_1, T_2, \dots, T_m\}$ be subsets of items going to machines $\{1, 2, \dots, m\}$ respectively. In this case, we say that $\{T_1, T_2, \dots, T_m\}$ is a *random clustering of $\mathbb{N}$ with multiplicity C* , i.e., $\{T_1, T_2, \dots, T_m\}$ is a family of subsets $T_i \subseteq \mathbb{N}$, where each item of $\mathbb{N}$ is assigned to C randomly chosen subsets in this family. Note that T_i 's are not necessarily disjoint subsets of items. Only the case of $C = 1$ corresponds to a random partitioning of items into m disjoint parts. This case is the most natural way of applying this idea, and is studied in Section 2. As we see later, higher values of C can help us achieve better approximation factors (See Section 3). We are now ready to formally define randomized composable core-sets.

DEFINITION 1.1. Consider an algorithm *ALG* that given any subset $T \subseteq \mathbb{N}$ returns a subset $\text{ALG}(T) \subseteq T$ with size at most k' . Let $\{T_1, T_2, \dots, T_m\}$ be a random clustering of $\mathbb{N}$ to m subsets with multiplicity C . We say that algorithm *ALG* is an algorithm that implements α -approximate randomized composable core-set of size k' with multiplicity C for f and cardinality constraint parameter k , if,

$$\mathbb{E}[f_k(\text{ALG}(T_1) \cup \dots \cup \text{ALG}(T_m))] \geq \alpha \cdot \mathbb{E}[f_k(T_1 \cup \dots \cup T_m)],$$

where the expectation is taken over the random choice of $\{T_1, T_2, \dots, T_m\}$. For brevity, instead of saying that *ALG* implements a composable core-set, we say that *ALG* is an α -approximate randomized composable core-set.

¹ While the concepts in this paper can be applied to other set functions, we focus on maximizing submodular set functions.

For ease of notation, when it is clear from the context, we may drop the term composable, and refer to composable core-sets as core-sets. Throughout this paper, we discuss randomized composable core-sets for the submodular maximization problem with a cardinality constraint k .

Distributed Approximation Algorithm. Note that we can use a randomized α -approximate composable core-set algorithm *ALG* to design the following simple distributed $(1 - \frac{1}{e})\alpha$ -approximation algorithm for monotone submodular maximization:

1. In the first phase, following the random clustering $\{T_1, \dots, T_m\}$ defined above, allocate items in $\mathbb{N}$ to m machines, i.e., machine i gets the subset T_i of items.
2. Each machine i computes a randomized composable core-set $S_i \subseteq T_i$ of size k' , i.e., $S_i = \text{ALG}(T_i)$ for each $1 \leq i \leq m$.
3. In the second phase, first collect the union of all core-sets, $U = \cup_{1 \leq i \leq m} S_i$, on one machine. Then apply a *post-processing* $(1 - \frac{1}{e})$ -approximation algorithm (e.g., algorithm *Greedy*) to compute a solution S to the submodular maximization problem over the set U . Output S .

It follows from the definition of the α -approximate randomized composable core-set that the above algorithm is a distributed $(1 - \frac{1}{e})\alpha$ -approximation algorithm for submodular maximization problem. We refer to this two-phase algorithmic approach as *the distributed algorithm*, and the overall approximation factor of the distributed algorithm as *the distributed approximation factor*. For all our algorithms in this paper, in addition to presenting an algorithm that achieves an approximation factor α as a randomized composable core-set, we propose a post-processing algorithm for the second phase, and present an improved analysis that achieves much better than $(1 - \frac{1}{e})\alpha$ -approximation as the distributed approximation factor.

Note that the above algorithm can be implemented in a distributed manner only if k' is small enough such that mk' items can be processed on one machine. In all our results the size of the composable core-set, k' , is a function of the cardinality constraint, k : In particular, in Section 2, we apply a composable core-set of size $k' = k$. In Section 3, we apply a composable core-set of size $k' < 4k$, and as a result, achieve a better approximation factor. We call a core-set, a *small-size core-set*, if its size k' is less than k (See full version of the paper).

Non-randomized Composable Core-sets. The above definition for randomized composable core-sets is introduced in this paper. Prior work [15, 8] define a non-randomized variant of composable core-sets where the above property holds for any (arbitrary) partitioning $\{T_1, T_2, \dots, T_m\}$ of data into m parts², i.e., an algorithm *ALG* as described above is a α -approximate (non-randomized) composable core-set of size k' for f , if for any cardinality constraint k , and any arbitrary partitioning $\{T_1, T_2, \dots, T_m\}$ of the items into m sets, we have $f_k(\text{ALG}(T_1) \cup \dots \cup \text{ALG}(T_m)) \geq \alpha \cdot f_k(T_1 \cup \dots \cup T_m)$.

² It is not hard to see that for non-randomized composable core-set, the multiplicity parameter C is not relevant.

1.2 Applications and Motivations

An α -approximate randomized composable core-set of size $k' = O(k)$ for a problem can be applied in three types of applications [15]³: (i) in distributed computation [13], where it implies an α -approximation in one or two rounds of MapReduces using the total communication complexity of $O(n)$, (ii) in the random-order streaming model, where it implies an α -approximation algorithm in one pass using sublinear memory, (iii) in a class of approximate nearest neighbor search problems, where it implies an α -approximation algorithm based on the locally sensitive hashing (under an assumption). Here, we discuss the application for the MapReduce framework, and for details of the other two applications, we refer to the full version and to [15]. We show how to use a randomized composable core-set of size $O(k)$ to design a distributed algorithm in one or two rounds of MapReduces⁴ using linear total communication complexity: Let $m = \sqrt{n/k}$, and let $(T_1, \dots, T_m)$ be a random partitioning where T_i has $\sqrt{kn}$ items. In the distributed algorithm, we assume that the random partitioning is produced in one round of MapReduce where each of m reducers receives T_i as input, and produces a core-set S_i for the next round. Alternatively, we may assume that the data (or the items) are distributed uniformly at random among machines, or similarity each of m mappers receives T_i as input, and produces a core-set S_i for the reducer. In either case, the produced core-sets are passed to a single reducer in the first or the second round. The total input to the reducer, i.e., the union of the core-sets, is of size at most $mk' = O(k)\sqrt{n/k} = O(\sqrt{kn})$. The solution computed by the reducer for the union of the core-sets is, by definition, a good approximation to the original problem. It is easy to see that the total communication complexity of this algorithm is $O(n)$, and this computation can be performed in one or two rounds as formally defined in the MapReduce computation model [17].

Variants of this technique have been applied for optimization under MapReduce framework [17, 21, 7, 15, 4, 8]. However, none of these previous results formally study the difference between randomized and non randomized variants and in most cases, they employ non-randomized composable core-sets. Indyk et al. [15] observed that the idea of non-randomized composable core-sets cannot be applied to the coverage maximization (or more generally submodular maximization) problems. In fact, all our hardness results also apply to a class of submodular maximization problems known as the maximum k -coverage problems, i.e., given a number k , and a family of subsets $\mathcal{A} \subset 2^X$, find a subfamily of k subsets $A_1, \dots, A_k$ whose union $\bigcup_{j=1}^k A_j$ is maximized. Solving max k -coverage and submodular maximization in a distributed manner have attracted a significant amount of research over the last few years [11, 12, 21, 9, 20, 22]. Other than the importance of these problems, one reason for the popularity of this problem in this context is the fact that its approximation algorithm is algorithm **Greedy** which is naturally sequential and it is hard to parallelize or implement in a distributed manner.

³These results assume $k \leq n^{1-\epsilon}$ for a constant ϵ .

⁴The straightforward way of applying the ideas will result in two rounds of MapReduce. However, if we assume that the data is originally sharded randomly and each part is in a single shard, and the memory for each machine is more than the size of each shard, then it can be implemented via one round of MapReduce computation.

1.3 Our Contributions

Our results are summarized in Table 1. As our first result, we prove that a family of efficient algorithms including a variant of algorithm **Greedy** with a consistent tie-breaking rule leads to an almost $1/3$ -approximate randomized composable core-set of size k for any monotone submodular function and cardinality constraint k with multiplicity of 1 (see Section 2). This is in contrast to a known $O(\frac{\log k}{\sqrt{k}})$ hardness result for any (non-randomized) composable core-set [15], and shows the advantage of using the randomization here. Furthermore, by constructing this randomized core-set and applying algorithm **Greedy** afterwards, we show a 0.27 distributed approximation factor for the monotone submodular maximization problem in *one or two rounds* of MapReduces with a *linear communication complexity*. Previous results lead to algorithms with either much larger number of rounds of MapReduce [11, 9], and/or larger communication complexity [20]. This improvement is important, since the number of rounds of MapReduce computation and communication complexity are the most important factors in determining the performance of a MapReduced-based algorithm [18]. The effectiveness of using this technique has been confirmed empirically by Mirzasoleiman et al [22] who studied a similar algorithm on a subclass of submodular maximization problems. However, they only provide provable guarantees for a subclass of submodular functions satisfying a certain Lipschitz condition [22]. Our result not only works for monotone submodular functions, but also *extends to non-monotone (non-negative) submodular functions*, and leads to the first constant-round MapReduce-based constant-factor approximation algorithm for non-monotone submodular maximization (with $O(n)$ total communication complexity and approximation factor of 0.18). It also leads to the first constant-factor approximation algorithm for non-monotone submodular maximization in a random-order streaming model in one pass with sublinear memory.

Our next goal in the paper is to improve the approximation factor of the above algorithm for monotone submodular functions. To this end, we first observe that one cannot achieve a better than the $1/2$ factor via core-sets of size k using algorithm **Greedy** or any algorithm in a family of local search algorithms. In Section 3, we show how to go beyond the $1/2$ -approximation by applying core-sets of size higher than k but still of size $O(k)$, and prove that algorithm **Greedy** with a consistent tie-breaking rule provides a 0.585-approximate randomized composable core-set of size $k' < 4k$ for our problem. We then present algorithm **PseudoGreedy** that can be applied as a post-processing step to design a distributed 0.545-approximation algorithm in one or two rounds of MapReduces, and with linear total communication complexity. For monotone submodular maximization, this result implies the first distributed approximation algorithm with approximation factor better than $1/2$ that runs in a constant number of rounds. We achieve this approximation factor using one or two rounds of MapReduces and with the total communication complexity of $O(n)$. In addition, this result implies the first approximation algorithm beating the $1/2$ factor for the random-order streaming model with constant number of passes on the data and sublinear memory. To complement this result, we first show that our analysis for algorithm **Greedy** is tight. Moreover, we show that it is information theoretically impossible to achieve an

Problem	Core-set Size	R/N	U/L	Core-set Approx. Factor	Distributed Approx.
Mon. Submodular Max. [15]	$\text{poly}(k)$	N	U	$O(\frac{\log k}{\sqrt{k}})$	-
Mon. Submodular Max.*	k	R	L	$1/3$	0.27
Non-Mon. Submodular Max.*	k	R	L	$\max(\frac{m-1}{3m}, \frac{1}{em}) \geq 0.18$	$\max(\frac{1-\frac{1}{m}}{2+e}, \frac{1}{em}) \geq 0.14$
Mon. Submodular Max.*	$O(k)$	R	L	$0.585 - O(\frac{1}{k})$	$0.545 - O(\frac{1}{k})$
Mon. Submodular Max.	$\text{poly}(k)$	R	U	$1 - \frac{1}{e}$	-
Mon. Submodular Max.	$k' < k$	N	UL	$\Theta(\frac{k'}{k})$	$\Theta(\frac{k'}{k})$
Mon. Submodular Max.	$k' < k$	R	UL	$\Theta(\sqrt{\frac{k'}{k}})$	$\Theta(\sqrt{\frac{k'}{k}})$

Table 1: This table summarizes our results. In the column titled "R/N", "R" corresponds to the randomized core-set, and "N" corresponds to the non-randomized core-set notion. In the column titled "U/L", "U" corresponds to an upper bound result, and "L" corresponds to a lower bound result. The last column corresponds to the distributed approximation factor. All these results except the first row are the new results of this paper. Previously, no constant-factor approximation has been proved for a randomized composable core-set for this problem. See Section 1.4 for comparison to previous approximation algorithms. The rows with a star(*) are our most important results.

approximation factor better than $1 - 1/e$ using a core-set with size polynomial in k .

Finally, we consider the construction of *small-size* core-sets, i.e., a core-set of size $k' < k$. Studying such core-sets is important particularly for cases with large parameter k , e.g., $k = \Omega(n)$ or $k = \frac{n}{\log n}$ ⁵. For our problem, we first observe a hardness bound of $O(\frac{k'}{k})$ for non-randomized core-sets. On the other hand, in the full version we show an $\Omega(\sqrt{\frac{k'}{k}})$ -approximate randomized composable core-set for this problem, and accompany this result by a matching hardness bound of $O(\sqrt{\frac{k'}{k}})$ for randomized composable core-set. The hardness result is presented in the full version.

1.4 More Related work.

Submodular Maximization in Streaming and MapReduce: Solving max k -coverage and submodular maximization in a distributed manner have attracted a significant amount of research over the last few years [11, 12, 21, 9, 20, 22, 5]. From theoretical point of view, for the coverage maximization problem, Chierchetti et al. [11] present a $(1 - 1/e)$ -approximation algorithm in polylogarithmic number of MapReduce rounds, and Bellocch et al [9] improved this result and achieved $\log^2 n$ number of rounds. Recently, Kumar et al. [20] present a $(1 - 1/e)$ -approximation algorithm using a logarithmic number of rounds of MapReduce. They also derive $(1/2 - \epsilon)$ -approximation algorithm that runs in $O(\frac{1}{\delta})$ number of rounds of MapReduce (for a constant δ), but this algorithm needs a $\log n$ blowup in the communication complexity. As observed in various empirical studies [18], the communication complexity and the number of MapReduce rounds are important factors in determining the performance of a MapReduce-based algorithm and a $\log n$ blowup in the communication complexity can play a crucial role in applicability of the algorithm in practice. Our algorithm on the other hand runs only in (one or) two rounds, and can run on any number of machines as long as they can store the data, i.e, it needs m machines each

⁵For such large k , a core-set of size k may not be as useful since outputting the whole core-set may be impossible. For example, in the formal MapReduce model [17], outputting a core-set of size k for $k = \Omega(n)$ is not feasible.

with memory proportional to $\frac{1}{m}$ of the size of the input. One previous attempt to apply the idea of core-sets for submodular maximization is by Indyk et al. [15] who rule out the applicability of non-randomized core-sets by showing a hardness bound of $O(\frac{\log k}{\sqrt{k}})$ for non-randomized core-sets. The most relevant previous attempt in applying randomized core-sets to submodular maximization is by Mirzasoleiman et al [22], where the authors study a class of algorithms similar to the ones discussed here, and show the effectiveness of applying algorithm **Greedy** over a random partitioning empirically for several machine learning applications. The authors also prove theoretical guarantees for algorithm **Greedy** for special classes of submodular functions satisfying a certain Lipschitz condition [22]. Here, on the other hand, we present guaranteed approximation results for all monotone and non-monotone submodular functions. Finally, there is a recent paper in the streaming model [5] in which the authors present a streaming $1/2$ -approximation algorithm with one pass and linear memory. Our results lead to improved results for random-order streaming model for monotone and non-monotone submodular maximization.

Core-sets: The notion of core-sets has been introduced in [3]. In this paper, we use the term core-sets to refer to "composable core-sets" which was formally defined in a recent paper [15]. This notion has been also implicitly used in Section 5 of Agarwal et al. [3] where the authors specify composability properties of ϵ -kernels (a variant of core-sets). The notion of (composable) core-sets are also related to the concept of mergeable summaries that have been studied in the literature [2]. As discussed before, the idea of using core-sets has been applied either explicitly or implicitly in the streaming model [14, 3] and in the MapReduce framework [17, 21, 7, 15, 8]. Moreover, notions similar to randomized core-sets have been studied for random-order streaming models [16].

1.5 More Notation

We end this section by presenting notations and definitions used in the rest of the paper. Let $\Delta(x, X)$ denote the marginal f value of adding item x to set X , i.e., $\Delta(x, X) \stackrel{\text{def}}{=} f(X \cup \{x\}) - f(X)$. With this definition, the submodularity property is equivalent to saying that for any two subsets

$X \subseteq Y \subseteq \mathbb{N}$ and an item $x \in \mathbb{N} \setminus Y$, we have $\Delta(x, X) \geq \Delta(x, Y)$. Consider the distributed algorithm described in Section 1.1. For any random clustering $\{T_1, \dots, T_m\}$, we let S_i ($1 \leq i \leq m$) be the output of algorithm **ALG** on set T_i , i.e., $S_i = \text{ALG}(T_i)$. Also we let OPT be the optimum solution ($\arg \max_{|S| \leq k} f(S)$), and OPT^S be the set of selected items of OPT in $\cup_{i=1}^m S_i$, i.e., $\text{OPT}^S \stackrel{\text{def}}{=} \text{OPT} \cap (\cup_{i=1}^m S_i)$. Throughout this paper, we analyze a class of algorithms, referred to as β -nice algorithms, with certain properties defined below:

DEFINITION 1.2. Consider a submodular set function f . Let **ALG** be an algorithm that given any subset T returns subset $\text{ALG}(T) \subseteq T$ with size at most k' . We say that this Algorithm **ALG** is a β -nice algorithm for function f and some parameter β iff for any set T and any item $x \in T \setminus \text{ALG}(T)$ (item x is in set T but is not selected in the output of algorithm **ALG**), then the following two properties hold:

- Set $\text{ALG}(T \setminus \{x\})$ is equal to $\text{ALG}(T)$, i.e., intuitively the output of the algorithm should not depend on the items it does not select, and
- $\Delta(x, \text{ALG}(T))$ is at most $\beta \frac{f(\text{ALG}(T))}{k'}$. In other words, the marginal f value of any not-selected item cannot be more than β times the average contribution of selected items.

2. RANDOMIZED CORE-SETS FOR SUBMODULAR MAXIMIZATION

In this section, we show that a family of β -nice algorithms, introduced in Section 1.5, leads to a constant-factor approximate randomized composable core-set, and a constant-factor distributed approximation algorithm for monotone and non-monotone submodular maximization problems with cardinality constraints. In the full version, we show that several efficient algorithms in the literature of submodular maximization are β -nice for some $\beta \in [1, 1 + \epsilon]$ (for $\epsilon = o(1)$) including some variant of algorithm **Greedy** with a consistent tie-breaking rule, and also an almost linear-time algorithm in [6]. Before stating the theorem, we emphasize that in this section, we apply a composable core-set of multiplicity 1 which corresponds to a *random partitioning* of items into m disjoint pieces.

THEOREM 2.1. For any $\beta > 0$, any β -nice algorithm **ALG** is a $\frac{1}{2+\beta}$ -approximate randomized composable core-set of multiplicity 1 and size k for the monotone, and $\frac{1-\frac{1}{m}}{2+\beta}$ -approximate for non-monotone submodular maximization problems with cardinality constraint k .

PROOF. We want to show that there exists a subset of $\cup_{i=1}^m S_i$ with size at most k , and at least an expected f value of $\frac{f(\text{OPT})}{2+\beta}$ for monotone f , and $\frac{(1-1/m)f(\text{OPT})}{2+\beta}$ for non-monotone f . Toward this goal, we take the maximum of $\max_{1 \leq i \leq m} f(S_i)$ and $f(\text{OPT}^S)$ as a candidate solution. We define some notation to simplify the rest of the proof. Consider an arbitrary permutation π on items of OPT , and for each item $x \in \text{OPT}$ let OPT^x be the set of items in π that appear before x . We will first lower bound $f(\text{OPT}^S)$ using the submodularity property in Lemma 2.2.

LEMMA 2.2. For any set of selected items, $f(\text{OPT}^S) \geq f(\text{OPT}) - \sum_{x \in \text{OPT} \setminus (\cup_{i=1}^m S_i)} \Delta(x, \text{OPT}^x)$ for a monotone or non-monotone submodular function f .

PROOF. First we note that $f(\text{OPT}) - f(\text{OPT}^S) = \sum_{x \in \text{OPT} \setminus \text{OPT}^S} \Delta(x, \text{OPT}^x \cup \text{OPT}^S)$. Using submodularity property, we know that $\Delta(x, \text{OPT}^x \cup \text{OPT}^S)$ is at most $\Delta(x, \text{OPT}^x)$ because OPT^x is a subset of $\text{OPT}^x \cup \text{OPT}^S$. So $f(\text{OPT}) - f(\text{OPT} \cap A) \leq \sum_{x \in \text{OPT} \setminus \text{OPT}^S} \Delta(x, \text{OPT}^x)$ which is equal to:

$$\sum_{x \in \text{OPT} \setminus (\cup_{i=1}^m S_i)} \Delta(x, \text{OPT}^x)$$

by definition of OPT^S . This concludes the proof. $\square$

We note that this technique of writing value of optimum in terms of Δ values (in proof of Lemma 2.2) is a simple and useful technique which has been introduced in another context [19]. Lemma 2.2 suggests that we should upper bound

$$\sum_{x \in \text{OPT} \setminus (\cup_{i=1}^m S_i)} \Delta(x, \text{OPT}^x)$$

which is done in the next lemma.

LEMMA 2.3. The following inequality holds

$$\begin{aligned} \sum_{i=1}^m \sum_{x \in \text{OPT} \cap T_i \setminus S_i} \Delta(x, \text{OPT}^x) &\leq \beta \left(\max_{1 \leq i \leq m} f(S_i) \right) \\ &+ \sum_{i=1}^m \sum_{x \in \text{OPT} \cap T_i \setminus S_i} (\Delta(x, \text{OPT}^x) - \Delta(x, \text{OPT}^x \cup S_i)) \end{aligned}$$

for a monotone or non-monotone submodular function f .

PROOF. The sum $\sum_{i=1}^m \sum_{x \in \text{OPT} \cap T_i \setminus S_i} \Delta(x, \text{OPT}^x)$ can be written as:

$$\begin{aligned} \sum_{i=1}^m \sum_{x \in \text{OPT} \cap T_i \setminus S_i} [\Delta(x, \text{OPT}^x \cup S_i) + \\ (\Delta(x, \text{OPT}^x) - \Delta(x, \text{OPT}^x \cup S_i))] \end{aligned}$$

The first term in the sum is upper bounded by $\beta \frac{f(S_i)}{k}$ using the second property of β -nice algorithms. To conclude the proof, we note that

$$f(S_i) \leq \max_{1 \leq i' \leq m} f(S_{i'}),$$

and there are at most k items in $\text{OPT} \setminus \text{OPT}^S = \cup_{i=1}^m (\text{OPT} \cap T_i \setminus S_i)$. $\square$

At this stage of the analysis, we use randomness of partition $\{T_i\}_{i=1}^m$ to upper bound the expected value of these differences in Δ values with the expected value of average of $f(S_i)$. This is stated in the following lemma:

LEMMA 2.4. Assuming each item in OPT is assigned to a T_i uniformly at random (which is the case for all items not only members of OPT), we have that

$$\mathbb{E} \left[\sum_{i=1}^m \sum_{x \in \text{OPT} \cap T_i \setminus S_i} \Delta(x, \text{OPT}^x) - \Delta(x, \text{OPT}^x \cup S_i) \right]$$

is at most $\frac{\mathbb{E}[\sum_{i=1}^m f(S_i)]}{m}$ for a monotone submodular function f , and at most $\frac{\mathbb{E}[\sum_{i=1}^m f(S_i)]}{m} + \frac{f(\text{OPT})}{m}$ for a non-monotone submodular function f .

Before proving the lemmas, we observe that putting these three lemmas together, we can finish the proof of the theorem. In particular, for a monotone or non-monotone submodular function f , we have that:

$$\begin{aligned} & \mathbb{E}[f(\text{OPT}^S)] \\ & \geq f(\text{OPT}) - \beta \mathbb{E}[\max_{1 \leq i \leq m} f(S_i)] \\ & \quad - \frac{\mathbb{E}[\sum_{i=1}^m f(S_i)]}{m} - \frac{f(\text{OPT})}{m} \\ & \geq (1 - \frac{1}{m})f(\text{OPT}) - (1 + \beta)\mathbb{E}[\max_{1 \leq i \leq m} f(S_i)]. \end{aligned}$$

This immediately implies that

$$\mathbb{E}[\max\{f(\text{OPT}^S), \max_{1 \leq i \leq m} f(S_i)\}] \geq \frac{(1 - \frac{1}{m})f(\text{OPT})}{2 + \beta},$$

which concludes the proof for non-monotone functions. For monotone functions, we get the same proof except that we should exclude the $-\frac{f(\text{OPT})}{m}$ term. To complete the proof, it is sufficient to prove the lemmas. The proofs of the first two lemmas are straightforward and are left to the full version. Here we prove Lemma 2.4 to elaborate on some of the techniques we use.

Proof of Lemma 2.4. The main part of the proof is to show that the sum of the Δ differences in the statement of the lemma is in expectation at most $\frac{1}{m}$ fraction of sum of Δ differences for a larger set of pairs (i, x) . In particular, we show that

$$\begin{aligned} & \mathbb{E}[\sum_{i=1}^m \sum_{x \in \text{OPT} \cap T_i \setminus S_i} \Delta(x, \text{OPT}^x) - \Delta(x, \text{OPT}^x \cup S_i)] \\ & \leq \frac{1}{m} \cdot \mathbb{E}[\sum_{i=1}^m \sum_{x \in \text{OPT}} \Delta(x, \text{OPT}^x) - \Delta(x, \text{OPT}^x \cup S_i)]. \end{aligned}$$

To simplify the rest of the proof, let A be the left hand side of the above inequality, and B be its right hand side. Also to simplify expressions A and B , we introduce the following notation: For every item x and set $T \subseteq \mathbb{N}$, let $h(x, T)$ denote $\Delta(x, \text{OPT}^x) - \Delta(x, \text{OPT}^x \cup \text{ALG}(T))$. Also let $\mathbb{1}[x \notin \text{ALG}(T \cup \{x\})]$ be equal to one if x is not in set $\text{ALG}(T \cup \{x\})$, and zero otherwise. We note that A and B are both separable for different choices of item x and set T_i , and can be rewritten formally using the new notation as follows:

$$\begin{aligned} A &= \sum_{i=1}^m \sum_{x \in \text{OPT}} \sum_{T \subseteq \mathbb{N} \setminus \{x\}} Pr[T_i = T \cup \{x\}] \\ & \quad \mathbb{1}[x \notin \text{ALG}(T \cup \{x\})] h(x, T \cup \{x\}) \\ B &= \sum_{i=1}^m \sum_{x \in \text{OPT}} \sum_{T \subseteq \mathbb{N} \setminus \{x\}} (Pr[T_i = T \cup \{x\}] \\ & \quad h(x, T \cup \{x\}) + Pr[T_i = T] h(x, T)) \\ & \geq \sum_{i=1}^m \sum_{x \in \text{OPT}} \sum_{T \subseteq \mathbb{N} \setminus \{x\}} \mathbb{1}[x \notin \text{ALG}(T \cup \{x\})] \\ & \quad h(x, T \cup \{x\}) (Pr[T_i = T \cup \{x\}] + Pr[T_i = T]) \end{aligned}$$

where the inequality is implied by the following simple observations. Function h is non-negative, so multiplying the sum by $\mathbb{1}[x \notin \text{ALG}(T \cup \{x\})]$ (which is either zero or one) can only decrease its value. We also replace one $h(x, T)$ with $h(x, T \cup \{x\})$ which does not change the value of the sum at all because when $\mathbb{1}[x \notin \text{ALG}(T \cup \{x\})] = 1$ (its only non-zero value), $\text{ALG}(T \cup \{x\})$ is identical to $\text{ALG}(T)$ using the first property of nice algorithms, and thus $h(x, T) = h(x, T \cup \{x\})$ by definition. So the inequality holds.

Now we can compare A , and B as follows: For any set $T \subseteq \mathbb{N} \setminus \{x\}$, we have:

$$Pr[T_i = T] = \left(\frac{1}{m}\right)^{|T|} \left(1 - \frac{1}{m}\right)^{|\mathbb{N}| - |T|}$$

and

$$Pr[T_i = T \cup \{x\}] = \left(\frac{1}{m}\right)^{|T|+1} \left(1 - \frac{1}{m}\right)^{|\mathbb{N}| - |T| - 1}$$

. As a result, the ratio $\frac{Pr[T_i = T \cup \{x\}]}{Pr[T_i = T \cup \{x\}] + Pr[T_i = T]}$ is equal to $\frac{1}{m}$ which shows that $A \leq B$.

To complete the proof for monotone f , it suffices to prove that $B \leq \frac{\mathbb{E}[\sum_{i=1}^m f(S_i)]}{m}$, we note that for any i , $\sum_{x \in \text{OPT}} \Delta(x, \text{OPT}^x \cup S_i) = f(\text{OPT} \cup S_i) - f(S_i)$, and $\sum_{x \in \text{OPT}} \Delta(x, \text{OPT}^x) = f(\text{OPT})$. For a monotone f , we have $f(\text{OPT} \cup S_i) \geq f(\text{OPT})$ which shows that $B \leq \frac{\mathbb{E}[\sum_{i=1}^m f(S_i)]}{m}$.

Although, we do not necessarily have $f(\text{OPT} \cup S_i) \geq f(\text{OPT})$ for a non-monotone f , we show that $\sum_{i=1}^m f(\text{OPT} \cup S_i) \geq (m-1)f(\text{OPT})$. For any $1 \leq i < m$, we have that $f(\text{OPT} \cup (\cup_{i'=1}^i S_{i'})) + f(\text{OPT} \cup S_{i+1})$ is at least $f(\text{OPT}) + f(\text{OPT} \cup (\cup_{i'=1}^{i+1} S_{i'}))$ by submodularity. Applying this inequality for each $1 \leq i < m$, and using non-negativity of f , we imply that $\sum_{i=1}^m f(\text{OPT} \cup S_i) \geq (m-1)f(\text{OPT})$. Therefore we have:

$$B \leq \frac{\mathbb{E}[\sum_{i=1}^m f(S_i)]}{m} + \frac{(m - (m-1))f(\text{OPT})}{m}$$

which completes the proof for non-monotone f as well.

For large enough m , the above result leads to a factor close to $1/3$ for non-monotone submodular functions. We elaborate in the full version to show that it is straightforward to achieve the approximation factor $\max(\frac{1}{3} - \frac{1}{3m}, \frac{1}{em})$ or 0.18 .

2.1 Overall Distributed Approximation Factor of β -nice Algorithms

In Theorem 2.1, we prove that if on each part (set T_i) of the partitioning, we run a β -nice algorithm ALG , the union of output sets of ALG will contain a set of size at most k that preserves at least $\frac{1}{2+\beta}$ fraction of value of optimum set. If we run algorithm Greedy on the union of output sets $\cup_{i=1}^m S_i$, using the classic analysis of algorithm Greedy , we can easily claim that the overall value of the output set at the end is at least $\frac{1-1/e}{2+\beta}$ fraction of $f(\text{OPT})$. In Theorem 2.5 (proved in the full version), we show an improved distributed approximation factor.

THEOREM 2.5. *Let S be the output of algorithm Greedy over $\cup_{i=1}^m S_i$, i.e. $S = \text{Greedy}(\cup_{i=1}^m S_i)$ for a monotone submodular function f . Also let S be the output of non-monotone*

submodular maximization algorithm of Buchbinder et al. [10] on $\cup_{i=1}^m S_i$ when f is a non-monotone submodular function. The expected value of $\max\{f(S), \max_{i=1}^m \{f(S_i)\}\}$ is at least $\frac{(1-1/e)f(\text{OPT})}{1+(1-1/e)(1+\beta)}$ for monotone and $\frac{(1-\frac{1}{m})f(\text{OPT})/e}{1+(1+\beta)/e}$ for non-monotone submodular f . In particular, for $\beta = 1$, the distributed approximation factors are ≥ 0.27 , and $\geq 0.21 - \frac{1}{4m}$ for monotone and non-monotone f respectively.

PROOF. By applying lemmas 2.2, 2.3, and 2.4, we have that: $f(\text{OPT}^S) \geq f(\text{OPT}) - \beta \max_{i=1}^m f(S_i) - \frac{\sum_{i=1}^m f(S_i)}{m} \geq f(\text{OPT}) - (1+\beta) \max_{i=1}^m f(S_i)$ for a monotone submodular function f . Using the classic analysis of Greedy on submodular maximization in [23], one can prove that $f(S) \geq (1-1/e)f(\text{OPT}^S)$ when f is monotone. By taking the expectation of the two sides of this inequality, and using Lemma 2.4, we have that:

$$\mathbb{E}[f(S)] \geq \mathbb{E}\left[(1-1/e) \left(f(\text{OPT}) - (1+\beta) \max_{i=1}^m f(S_i)\right)\right]$$

By letting $\rho := \frac{\mathbb{E}[\max\{f(S), \max_{i=1}^m \{f(S_i)\}\}]}{f(\text{OPT})}$, the above inequality implies that $\rho \geq (1-1/e)(1 - (1+\beta)\rho)$, and therefore $\rho \geq \frac{1-1/e}{1+(1-1/e)(1+\beta)}$. Consequently $\mathbb{E}[\max\{f(S), \max_{i=1}^m \{f(S_i)\}\}]$ is at least $\frac{(1-1/e)f(\text{OPT})}{1+(1-1/e)(1+\beta)}$ which proves the claim for any monotone f .

If f is non-monotone, we get a weaker inequality $f(S) \geq \frac{f(\text{OPT}^S)}{e}$ by applying the algorithm of Buchbinder et al. [10]. Using lemmas 2.2, 2.3, and 2.4, we have that: $f(\text{OPT}^S) \geq f(\text{OPT}) - \beta \max_{i=1}^m f(S_i) - \frac{\sum_{i=1}^m f(S_i)}{m} - \frac{f(\text{OPT})}{m} \geq (1 - \frac{1}{m})f(\text{OPT}) - (1+\beta) \max_{i=1}^m f(S_i)$. Similarly, we can claim that $\rho \geq \frac{(1-\frac{1}{m})/e}{1+(1+\beta)/e}$. This implies the desired lower bound of $\frac{(1-\frac{1}{m})/e}{1+(1+\beta)/e}$ on ρ . $\square$

Non-monotone submodular maximization In Theorem 2.1, we proved that for a non-monotone f , Greedy returns a $(\frac{1}{3} - \frac{1}{3m})$ -approximate randomized composable core-set. When we have a small number of machines, m , this approximation factor becomes small. So we suggest the non-monotone submodular maximization algorithm of Buchbinder et al. [10] as an alternative. We note that since the items are partitioned randomly, $\frac{1}{m}$ fraction of optimum solution is sent to each machine in expectation, and the expected value $\mathbb{E}[f(\text{OPT} \cap S_i)]$ is at least $\frac{f(\text{OPT})}{m}$ for any $1 \leq i \leq m$. Using the algorithm of Buchbinder et al. we get a $\frac{1}{em}$ -approximate randomized core-set. We also note that the 0.21-approximation guarantee for non-monotone f in Theorem 2.5 holds for large enough number of machines m . For small m , this $\frac{1}{em}$ approximation can be used as an alternative.

2.2 Examples of β -Nice Algorithms

In this section, we show that several existing algorithms for submodular maximization in the literature belong to the family of β -nice algorithms.

Algorithm Greedy with Consistent Tie-breaking: First, we observe that algorithm Greedy is 1-nice if it has a consistent tie breaking rule: while selecting among the items with the same marginal value, Greedy can have a fixed strict total ordering (Π) of the items, and among the set of items with the maximum marginal value chooses the one highest rank in Π . The consistency of the tie breaking rule implies the

first property of nice algorithms. To see the second property, first observe that (i) Greedy always adds an item with the maximum marginal f value, and (ii) using submodularity of f , the marginal f values are decreasing as more items are added to the selected items. Therefore, after k iterations, the marginal value of adding any other item, is less than each of the k marginal f values we achieved while adding the first k items. This implies the 2nd property, and concludes that Greedy with a consistent tie-breaking rule is a 1-nice algorithm.

An almost linear-time $(1+\epsilon)$ -nice Algorithm:

Badanidiyuru and Vondrak [6] present an almost linear-time $(1 - \frac{1}{e} - \epsilon)$ -approximation algorithm for monotone submodular maximization with a cardinality constraint. We observe that this algorithm is $(1+2\epsilon)$ -nice. The algorithm is a relaxed version of Greedy where in each iteration, it adds an item with almost maximum marginal value (with at least $1 - \epsilon$ fraction of the maximum marginal). As a result, similar to the proof for Greedy, one can show that this linear-time algorithm is $\frac{1}{1-\epsilon}$ -nice and consequently $(1+2\epsilon)$ -nice for $\epsilon \leq 0.5$.

3. BETTER RANDOMIZED CORE-SETS FOR MONOTONE SUBMODULAR MAXIMIZATION

In full version of the paper, we show that if we restrict our attention to core-sets of size k , algorithm Greedy or a family of local search algorithms do not achieve an approximation factor better than $\frac{1}{2}$ even if each item is sent to multiple machines (up to multiplicity $C = o(\sqrt{m})$). This leads to the following question: does increasing the output size of core-sets, k' , help with the approximation factor? In other words, can we get a better than $1/2$ approximation factor if we allow the algorithm to select more than k items on each machine? To answer this question, we first show in the full version that it is not possible to achieve a randomized composable core-set of size $k' = o(\frac{n}{Cm})$ with approximation better than $1 - \frac{1}{e}$ even when we allow for multiplicity $C = o(\sqrt{\frac{m}{k}})$. Here, we show that although it is not possible to beat the $1 - \frac{1}{e}$ barrier, we can slightly increase the output sizes, apply the algorithm Greedy for the output size of $k' = (\sqrt{2} + 1)k$ and achieve an approximation factor $\approx 2 - \sqrt{2} > \frac{1}{2}$ with a constant multiplicity. Furthermore, we show in the full version that our analysis is tight for algorithm Greedy even if we increase the core-set sizes significantly. Since in this section, we are dealing with a monotone submodular function f , we can assume WLOG that $f(\emptyset) = 0$. Here, we present the main theorem and its proof sketch, and leave the proof of its main lemmas to the full version.

THEOREM 3.1. *For any integer $C \geq 1$, any cardinality constraint $k = o(m)$, algorithm Greedy is a $(2 - \sqrt{2} - O(\frac{1}{k} + \frac{\ln(C)}{C}))$ -approximate randomized composable core-set of multiplicity C and size $k' = (2\sqrt{2} + 1)k$ for any monotone submodular function f . By letting $C = \frac{1}{\epsilon}$, this leads to a randomized composable core-set of approximation factor 0.5857.*

PROOF. Let $D \stackrel{\text{def}}{=} 2\sqrt{2} + 1$, and $k' = Dk$. Following our notation from Section 1.5, let $S_i \stackrel{\text{def}}{=} \text{Greedy}(T_i)$ for $1 \leq i \leq m$, where T_i is the set of items sent to the machine i . Note that we can let $|S_i| = Dk$, since if there are less than Dk

items in T_i , WLOG we can assume the algorithm returns some extra dummy items just for the sake of analysis.

Consider an item $x \in \text{OPT}$. We say that x *survives from machine i* , if, when we send x to machine i in addition to items of T_i , algorithm **Greedy** would choose this item x in its output of size k' , i.e., if $x \in \text{Greedy}(T_i \cup \{x\})$. For the sake of analysis, we partition the optimum solution into two sets as follows: let OPT_1 be the set of items in the optimum solution that would survive the first machine, i.e., $\text{OPT}_1 \stackrel{\text{def}}{=} \{x | x \in \text{OPT} \cap \text{Greedy}(T_1 \cup \{x\})\}$. Let $\text{OPT}_2 \stackrel{\text{def}}{=} \text{OPT} \setminus \text{OPT}_1$, and $k_1 \stackrel{\text{def}}{=} |\text{OPT}_1|$, and $k_2 \stackrel{\text{def}}{=} |\text{OPT}_2|$ (note that $k_1 + k_2 = k$). We also define $\text{OPT}'_1 \stackrel{\text{def}}{=} \text{OPT}_1 \cap \text{OPT}^S$, and $\text{OPT}' \stackrel{\text{def}}{=} \text{OPT}'_1 \cup \text{OPT}_2$ where OPT^S is defined in Subsection 1.5.

We aim to prove that $\mathbb{E}[f_k(\cup_{i=1}^m S_i)]$ is at least $(2 - \sqrt{2} - O(\frac{1}{k} + \frac{\ln(C)}{C}))f(\text{OPT})$. Since f_k is a monotone function, we have $f_k(S_1 \cup \text{OPT}'_1) \leq f_k(\cup_{i=1}^m S_i)$. Note that OPT'_1 is by definition a subset of OPT^S , and consequently a subset of $\cup_{i=1}^m S_i$. So it suffices to prove that $\mathbb{E}[f_k(S_1 \cup \text{OPT}'_1)]$ is at least $(2 - \sqrt{2} - O(\frac{1}{k} + \frac{\ln(C)}{C}))f(\text{OPT})$. To do so, we first prove the following Lemma:

LEMMA 3.2. *The expected value of set $\mathbb{E}[f(\text{OPT}')] is at least $(1 - O(\frac{\ln(C)}{C}))f(\text{OPT})$.$*

Using Lemma 3.2, it is sufficient prove that ratio $\frac{f_k(S_1 \cup \text{OPT}'_1)}{f(\text{OPT}')}$ is at least $2 - \sqrt{2} - O(\frac{1}{k})$. In order to lower bound the ratio $\frac{f_k(S_1 \cup \text{OPT}'_1)}{f(\text{OPT}')}$, we write the following factor-revealing linear program LP^{k,k_2} , and prove in Lemma 3.3 that the solution to this LP is a lower bound on the aforementioned ratio.

LP^{k,k_2} Minimize β Subject to:
(1) $\beta \geq 1 - \alpha + \sum_{j \in J} a_j + c_j \quad \forall J \subset [Dk] \& J = k_2$
(2) $a_j + b_j + c_j \geq \frac{\alpha - \sum_{j'=1}^{j-1} a_{j'}}{k_2} \quad \forall 1 \leq j \leq Dk$
(3) $\sum_{j=1}^{Dk} b_j \leq 1 - \alpha$
(4) $0 \leq \alpha, a_j, b_j, c_j \leq 1 \quad \forall 1 \leq j \leq Dk$
(5) $\beta \geq \sum_{j=1}^{k_2} a_j + b_j + c_j$
(6) $a_j + b_j + c_j \geq a_{j+1} + b_{j+1} + c_{j+1} \quad \forall 1 \leq j < Dk$

LEMMA 3.3. *For any integer $k > 0$, the ratio $\frac{f_k(S_1 \cup \text{OPT}'_1)}{f(\text{OPT}')}$ is lower bounded by the optimum solution of linear program LP^{k,k_2} for some integer $1 \leq k_2 \leq k$.*

Finally, we show that the solution of LP^{k,k_2} is at least $2 - \sqrt{2} - O(\frac{1}{k})$ for any possible value of k_2 which concludes the proof of this theorem.

LEMMA 3.4. *The optimum solution of linear program LP^{k,k_2} is at least $2 - \sqrt{2} - O(\frac{1}{k})$ for any $0 \leq k_2 \leq k$.*

□

Improved Distributed Approximation Algorithm: We remind that in the first phase, each machine $1 \leq i \leq m$ runs algorithm **Greedy** on set T_i with $k' = Dk$ (where D is $2\sqrt{2} + 1$). By Theorem 3.1, there exists a size k subset of selected items $\cup_{i=1}^m S_i$ with expected value at least $0.585f(\text{OPT})$, but we do not know how to find this set efficiently. If we apply algorithm **Greedy** again on $\cup_{i=1}^m S_i$ to select k items in total, we achieve a distributed approximation factor of

$(1 - \frac{1}{e}) \times 0.585 \approx 0.37$. In the following, we present a post-processing algorithm **PseudoGreedy** that achieves an overall distributed approximation factor better than $1/2$. In particular, after the first phase, we show how to find a size k subset of the union of selected items $\cup_{i=1}^m S_i$ with expected value at least $(0.545 - o(1))f(\text{OPT})$. Here we present the **PseudoGreedy** algorithm itself and leave the proof of this claim to the full version.

Algorithm **PseudoGreedy** proceeds as follows: it first computes a family of candidate solutions of size $k + O(1)$, and keeps the one candidate solution V with the maximum value. It then lets S to be a random size k subset of V , and returns S as the solution. These candidate solutions, denoted by $S_{k'_2, I}$ (for $1 \leq k'_2 \leq k$ and $4I \subseteq \{1, \dots, 8\}$) in Algorithm 1, are computed as follows: We first enumerate all k possible values of $1 \leq k'_2 \leq k$ (this notation is used to be consistent with the proof of Theorem 3.1). Then, by letting $k'_1 = k - k'_2$, and $k_3 = 32 \lceil \frac{k'_2}{128} \rceil$, we partition the first $8k_3$ items in set S_1 ⁶ into eight subsets $\{A_{i'}\}_{i'=1}^8$, each of size k_3 . The next step of the algorithm proceeds as follows: for any $I \subseteq \{1, 2, \dots, 8\}$ with $|I| \leq 4 + \frac{k'_1}{k_3}$, we initialize set $S_{k'_2, I}$ with the union of all sets $A_{i'}$ where $i' \in I$. Then for $k'_1 + (4 - |I|)k_3$ iterations, we search all items in $\cup_{i=1}^m S_i$, and insert the item with the maximum marginal value to $S_{k'_2, I}$. Roughly speaking, by starting from all these subsets, we ensure that the selected set hits enough number of items in OPT . The upper bound we enforce on $|I|$ is to make sure that the number of iterations at this step is non-negative, i.e. $k'_1 + (4 - |I|)k_3 \geq 0$. Finally we define V to be the set $S_{k'_2, I}$ with the maximum f value, and return a random subset of size k of V as the output set S .

```

1 Input: A collection of  $m$  subsets  $\{S_1, \dots, S_m\}$ .
2 Output: Set  $S \subset \cup_{i=1}^m S_i$  with  $|S| \leq k$ .
3  $V \leftarrow \emptyset$ ;
4 forall the  $1 \leq k'_2 \leq k$  do
5    $k_3 \leftarrow 32 \lceil \frac{k'_2}{128} \rceil$ ;
6    $k'_1 \leftarrow k - k'_2$ ;
7   Partition the first  $8k_3$  items of  $S_1$  into 8 sets
    $\{A_{i'}\}_{i'=1}^8$  each of size  $k_3$ ;
8   forall the  $I \subseteq \{1, 2, \dots, 8\}$  with  $|I| \leq 4 + \frac{k'_1}{k_3}$  do
9      $S_{k'_2, I} \leftarrow \cup_{i' \in I} A_{i'}$ ;
10    for  $k'_1 + (4 - |I|)k_3$  times do
11      Find  $\arg \max_{x \in \cup_{i=1}^m S_i} \Delta(x, S_{k'_2, I})$ , and insert
      it to  $S_{k'_2, I}$ ;
12    end
13    if  $f(S_{k'_2, I}) > f(V)$  then  $V \leftarrow S_{k'_2, I}$ ;
14  end
15 end
16  $S \leftarrow$  a random size  $k$  subset of  $V$ ;
Algorithm 1: Algorithm PseudoGreedy

```

THEOREM 3.5. *Algorithm **PseudoGreedy** returns a subset S with size at most k , and expected value at least $(0.545 - O(\frac{1}{k} + \frac{\ln(C)}{C}))f(\text{OPT})$.*

⁶ We choose any machine instead of machine 1, and since the clustering is done at random, the analysis goes through.

4. CONCLUSION

The concept of composable core-sets has been introduced recently in the context of distributed and streaming algorithms and have been applied to several problems [15, 8]. In this paper, we introduced the concept of *randomized* composable core-sets and showed its effectiveness in maximizing submodular functions in a distributed manner. While we mainly discuss the cardinality constraint in this paper, we expect that the ideas and the proof techniques be applicable to more general packing constraints such as matroid constraints. There are several research problems that are interesting to explore in this line of research.

- For the submodular maximization problem, it remains to find a randomized composable core-set of approximation factor $1 - \frac{1}{e}$, or rule out the possibility of constructing such a core-set.
- We discussed how the size and multiplicity of the composable core-set can help improve the approximation factor of the algorithms. It would be nice to get tight bounds on the approximation factor for each range of the size of composable core-sets. Moreover, it would be interesting to study the impact of increasing the multiplicity of the core-set on the achievable approximation factors.
- While we provided a tight result for the small-size composable core-set problem, the achievable approximation factor is not satisfactory. A natural way to improve this factor is to apply the composable core-set idea iteratively, and achieve an improved approximation factor. Even for composable core-sets of size k and above, it might be possible to improve the approximation factor by applying such a core-set multiple times. This approach leads to several interesting follow-up questions.
- While randomized composable core-sets are applicable to random-order streaming models, applying the proof techniques in this paper may result in stronger approximation factors in pure random-order streaming models (compared to the ones presented here). We leave this problem to future research.

Finally, it would be nice to explore applicability of this idea on other optimization and graph theoretic problems.

References

- [1] S. Abbar, S. Amer-Yahia, P. Indyk, S. Mahabadi, and K. R. Varadarajan. Diverse near neighbor problem. In *Symposium on Computational Geometry 2013, SoCG '13, Rio de Janeiro, Brazil, June 17-20, 2013*, pages 207–214, 2013.
- [2] P. K. Agarwal, G. Cormode, Z. Huang, J. Phillips, Z. Wei, and K. Yi. Mergeable summaries. In *Proceedings of the 31st symposium on Principles of Database Systems*, pages 23–34. ACM, 2012.
- [3] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *Journal of the ACM (JACM)*, 51(4):606–635, 2004.
- [4] A. Andoni, A. Nikolov, K. Onak, and G. Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 574–583, 2014.
- [5] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: Massive data summarization on the fly. In *KDD*, 2014.
- [6] A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *SODA*, pages 1497–1514, 2014.
- [7] M.-F. Balcan, S. Ehrlich, and Y. Liang. Distributed clustering on graphs. In *NIPS*, page to appear, 2013.
- [8] M. Bateni, A. Bhashkara, S. Lattanzi, and V. Mirrokni. Mapping core-sets for balanced clustering. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, 2014.
- [9] G. E. Blelloch, H. V. Simhadri, and K. Tangwongsan. Parallel and i/o efficient set covering algorithms. In *SPAA*, pages 82–90, 2012.
- [10] N. Buchbinder, M. Feldman, J. S. Naor, and R. Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '14*, pages 1433–1452. SIAM, 2014.
- [11] F. Chierichetti, R. Kumar, and A. Tomkins. Max-cover in map-reduce. In *WWW*, pages 231–240, 2010.
- [12] G. Cormode, H. J. Karloff, and A. Wirth. Set cover algorithms for very large datasets. In *CIKM*, pages 479–488, 2010.
- [13] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150, 2004.
- [14] S. Guha, N. Mishra, R. Motwani, and L. O’Callaghan. Clustering data streams. *STOC*, 2001.
- [15] P. Indyk, S. Mahabadi, M. Mahdian, and V. Mirrokni. Composable core-sets for diversity and coverage maximization. In *ACM PODS*, 2014.
- [16] M. Kapralov, S. Khanna, and M. Sudan. Approximating matching size from random streams. In *SODA*, pages 734–751, 2014.
- [17] H. J. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for mapreduce. In *SODA*, pages 938–948, 2010.
- [18] R. Kiveris, S. Lattanzi, V. Mirrokni, V. Rastogi, and S. Vasilvitski. Connected components in mapreduce and beyond. In *ACM SOCC*, 2014.
- [19] N. Korula, V. Mirrokni, and M. Zadimoghaddam. Online submodular welfare maximization: Greedy beats $1/2$ in random order. In *to appear STOC*, 2015.

- [20] R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani. Fast greedy algorithms in mapreduce and streaming. In *SPAA*, pages 1–10, 2013.
- [21] S. Lattanzi, B. Moseley, S. Suri, and S. Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In *SPAA*, pages 85–94, 2011.
- [22] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2049–2057, 2013.
- [23] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.