The Power of Randomization: Distributed Submodular Maximization on Massive Datasets

Rafael Barbosa[1] RAFAEL@DCS.WARWICK.AC.UK
Department of Computer Science and DIMAP, University of Warwick

Alina Ene A.ENE@DCS.WARWICK.AC.UK
Department of Computer Science and DIMAP, University of Warwick

Huy Le Nguyen HLNGUYEN@CS.PRINCETON.EDU
Simons Institute, University of California, Berkeley

Justin Ward J.D.WARD@DCS.WARWICK.AC.UK
Department of Computer Science and DIMAP, University of Warwick

Abstract

A wide variety of problems in machine learning, including exemplar clustering, document summarization, and sensor placement, can be cast as constrained submodular maximization problems. Unfortunately, the resulting submodular optimization problems are often too large to be solved on a single machine. We consider a distributed, greedy algorithm that combines previous approaches with randomization. The result is an algorithm that is embarrassingly parallel and achieves provable, constant factor, worst-case approximation guarantees. In our experiments, we demonstrate its efficiency in large problems with different kinds of constraints with objective values always close to what is achievable in the centralized setting.

1. Introduction

A set function $f : 2^V \to \mathbb{R}_{\geq 0}$ on a ground set V is *submodular* if $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$ for any two sets $A, B \subseteq V$. Several problems of interest can be modeled as maximizing a submodular objective function subject to certain constraints:

$$\max f(A) \text{ subject to } A \in \mathcal{C},$$

where $\mathcal{C} \subseteq 2^V$ is the family of feasible solutions. Indeed, the general meta-problem of optimizing a constrained submodular function captures a wide variety of problems in machine learning applications, including exemplar clustering, document summarization, sensor placement, image segmentation, maximum entropy sampling, and feature selection.

At the same time, in many of these applications, the amount of data that is collected is quite large and it is growing at a very fast pace. For example, the wide deployment of sensors has led to the collection of large amounts of measurements of the physical world. Similarly, medical data and human activity data are being captured and stored at an ever increasing rate and level of detail. This data is often high-dimensional and complex, and it needs to be stored and processed in a distributed fashion.

In these settings, it is apparent that the classical algorithmic approaches are no longer suitable and new algorithmic insights are needed in order to cope with these challenges. The algorithmic challenges stem from the following competing demands imposed by huge datasets: the computations need to process the data that is distributed across several machines using a minimal amount of communication and synchronization across the machines, and at the same time deliver solutions that are competitive with the centralized solution on the entire dataset.

The main question driving the current work is whether these competing goals can be reconciled. More precisely, can we deliver very good approximate solutions with minimal communication overhead? Perhaps surprisingly, the answer is yes; there is a very simple distributed greedy algorithm that is embarrassingly parallel and it achieves

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

[1]The authors are listed alphabetically.

provable, constant factor, worst-case approximation guarantees. Our algorithm can be easily implemented in a parallel model of computation such as MapReduce (Dean & Ghemawat, 2004).

1.1. Background and Related Work

In the MapReduce model, there are m independent machines. Each of the machines has a limited amount of memory available. In our setting, we assume that the data is much larger than any single machine's memory and so must be distributed across all of the machines. At a high level, a MapReduce computation proceeds in several rounds. In a given round, the data is shuffled among the machines. After the data is distributed, each of the machines performs some computation on the data that is available to it. The output of these computations is either returned as the final result or becomes the input to the next MapReduce round. We emphasize that the machines can only communicate and exchange data during the shuffle phase.

In order to put our contributions in context, we briefly discuss two distributed greedy algorithms that achieve complementary trade-offs in terms of approximation guarantees and communication overhead.

Mirzasoleiman *et al.* (2013) give a distributed algorithm, called GREEDI, for maximizing a monotone submodular function subject to a cardinality constraint. The GREEDI algorithm partitions the data arbitrarily on the machines and on each machine it then runs the classical GREEDY algorithm to select a feasible subset of the items assigned to that machine. The GREEDY solutions on these machines are then placed on a single machine and the GREEDY algorithm is used once more to select the final solution from amongst the resulting set of items. The GREEDI algorithm is very simple and embarrassingly parallel, but its worst-case approximation guarantee[2] is $1/\Theta(\min\{\sqrt{k}, m\})$, where m is the number of machines and k is the cardinality constraint. Mirzasoleiman *et al.* show that the GREEDI algorithm achieves very good approximations for datasets with geometric structure, and performs well in practice for a wide variety of experiments.

Kumar *et al.* (2013) give distributed algorithms for maximizing a monotone submodular function subject to a cardinality or more generally, a matroid constraint. Their algorithm combines the Threshold Greedy algorithm of (Gupta et al., 2010) with a sample and prune strategy. In each round, the algorithm samples a small subset of the elements that fit on a single machine and runs the Threshold Greedy algorithm on the sample in order to obtain a feasible solution. This solution is then used to prune some of the elements in the dataset and reduce the size of the ground set. The SAMPLE&PRUNE algorithms achieve constant factor approximation guarantees but they incur a higher communication overhead. For a cardinality constraint, the number of rounds is a constant but for more general constraints such as a matroid constraint, the number of rounds is $\Theta(\log \Delta)$, where Δ is the maximum increase in the objective due to a single element. The maximum increase Δ can be much larger than even the number of elements in the entire dataset, which makes the approach infeasible for massive datasets.

On the negative side, Indyk et al. (2014) studied coreset approaches to develop distributed algorithms for finding representative and yet diverse subsets in large collections. While succeeding in several measures, they also showed that their approach provably *does not* work for k-coverage, which is a special case of submodular maximization with a cardinality constraint.

1.2. Our Contribution

In this paper, we analyze a variant of the distributed GREEDI algorithm of (Mirzasoleiman et al., 2013), and show that one can achieve both the communication efficiency of the GREEDI algorithm and a provable, constant factor approximation guarantee. Our analysis relies crucially on the following modification: instead of partitioning the dataset arbitrarily onto the machines, we perform this initial partitioning *randomly*. Our analysis thus provides some theoretical justification for the very good empirical performance of the GREEDI algorithm that was established previously in the extensive experiments of (Mirzasoleiman et al., 2013). Moreover, we show that this approach delivers provably good performance in much wider settings than originally envisioned.

The GREEDI algorithm was originally studied in the special case of monotone submodular maximization under a cardinality constraint. In contrast, our analysis holds for any hereditary constraint. Specifically, we show that the randomized variant of the GREEDI algorithm achieves a constant factor approximation for any hereditary, constrained problem for which the classical (centralized) GREEDY algorithm achieves a constant factor approximation. This is the case not only for cardinality constraints, but also for matroid constraints, knapsack constraints, and p-system constraints (Jenkyns, 1976), which generalize the intersection of p matroid constraints. Table 1 gives the approximation ratio α obtained by the GREEDY algorithm on a variety of problems, and the corresponding constant factor obtained by the randomized GREEDI algorithm.

[2]Mirzasoleiman *et al.* (2013) give a family of instances where the approximation achieved is only $1/\min\{k, m\}$ if the solution picked on each of the machines is the optimal solution for the set of items on the machine. These instances are not hard for the GREEDI algorithm. We show in the supplement that the GREEDI algorithm achieves a $1/\Theta(\min\{\sqrt{k}, m\})$ approximation.

Table 1. New approximation bounds for randomized GREEDI for constrained monotone and non-monotone submodular maximization

Constraint	Centralized GREEDY		GREEDI Monotone	GREEDI Non-Monotone
cardinality	$1-\frac{1}{e}$	(Nemhauser et al., 1978)	$\frac{1}{2}(1-\frac{1}{e})$	$\frac{1}{10}$
matroid	$\frac{1}{2}$	(Fisher et al., 1978)	$\frac{1}{4}$	$\frac{1}{10}$
knapsack	≈ 0.35	(Wolsey, 1982)[3]	≈ 0.17	$\frac{1}{14}$
p-system	$\frac{1}{p+1}$	(Fisher et al., 1978)	$\frac{1}{2(p+1)}$	$\frac{1}{2+4(p+1)}$

Additionally, we show that if the greedy algorithm satisfies a slightly stronger technical condition, then our approach gives a constant factor approximation for constrained *non-monotone* submodular maximization. The resulting approximation ratios for non-monotone maximization problems are given in the last column of Table 1.

1.3. Preliminaries

MapReduce Model. In a MapReduce computation, the data is represented as $\langle \mathrm{key}, \mathrm{value}\rangle$ pairs and it is distributed across m machines. The computation proceeds in rounds. In a given round, the data is processed in parallel on each of the machines by *map tasks* that output $\langle \mathrm{key}, \mathrm{value}\rangle$ pairs. These pairs are then shuffled by *reduce tasks*; each reduce task processes all the $\langle \mathrm{key}, \mathrm{value}\rangle$ pairs with a given key. The output of the reduce tasks either becomes the final output of the MapReduce computation or it serves as the input of the next MapReduce round.

Submodularity. As noted in the introduction, a set function $f : 2^V \to \mathbb{R}_{\geq 0}$ is *submodular* if, for all sets $A, B \subseteq V$,

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

A useful alternative characterization of submodularity can be formulated in terms of diminishing marginal gains. Specifically, f is submodular if and only if:

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$$

for all $A \subseteq B \subseteq V$ and $e \notin B$.

The *Lovász extension* $f^- : [0,1]^V \to \mathbb{R}_{\geq 0}$ of a submodular function f is given by:

$$f^-(\mathbf{x}) = \mathop{\mathbb{E}}_{\theta \in \mathcal{U}(0,1)}[f(\{i : x_i \geq \theta\})].$$

For any submodular function f, the Lovász extension f^- satisfies the following properties: (1) $f^-(\mathbf{1}_S) = f(S)$ for all $S \subseteq V$, (2) f^- is convex, and (3) $f^-(c \cdot \mathbf{x}) \geq c \cdot f^-(\mathbf{x})$ for any $c \in [0,1]$. These three properties immediately give the following simple lemma:

[3]Wolsey's algorithm satisfies all technical conditions required for our analysis (in particular, those for Lemma 2).

Lemma 1. *Let S be a random set, and suppose that $\mathbb{E}[\mathbf{1}_S] = c \cdot \mathbf{p}$ (for $c \in [0,1]$). Then, $\mathbb{E}[f(S)] \geq c \cdot f^-(\mathbf{p})$.*

Proof. We have:

$$\begin{aligned}\mathbb{E}[f(S)] &= \mathbb{E}[f^-(\mathbf{1}_S)] \\ &\geq f^-(\mathbb{E}[\mathbf{1}_S]) = f^-(c \cdot \mathbf{p}) \geq c \cdot f^-(\mathbf{p}),\end{aligned}$$

where the first equality follows from property (1), the first inequality from property (2), and the final inequality from property (3). □

Hereditary Constraints. Our results hold quite generally for any problem which can be formulated in terms of a hereditary constraint. Formally, we consider the problem

$$\max\{f(S) : S \subseteq V, S \in \mathcal{I}\}, \tag{1}$$

where $f : 2^V \to \mathbb{R}_{\geq 0}$ is a submodular function and $\mathcal{I} \subseteq 2^V$ is a family of feasible subsets of V. We require that $\mathcal{I}$ be *hereditary* in the sense that if some set is in $\mathcal{I}$, then so are all of its subsets. Examples of common hereditary families include cardinality constraints ($\mathcal{I} = \{A \subseteq V : |A| \leq k\}$), matroid constraints ($\mathcal{I}$ corresponds to the collection independent sets of the matroid), knapsack constraints ($\mathcal{I} = \{A \subseteq V : \sum_{i \in A} w_i \leq b\}$), as well as combinations of such constraints. Given some constraint $\mathcal{I} \subseteq 2^V$, we shall also consider restricted instances in which we are presented only with a subset $V' \subseteq V$, and must find a set $S \subseteq V'$ with $S \in \mathcal{I}$ that maximizes f. We say that an algorithm is an α-approximation for maximizing a submodular function subject to a hereditary constraint $\mathcal{I}$ if, for any submodular function $f : 2^V \to \mathbb{R}_{\geq 0}$ and any subset $V' \subseteq V$ the algorithm produces a solution $S \subseteq V'$ with $S \in \mathcal{I}$, satisfying $f(S) \geq \alpha \cdot f(\mathrm{OPT})$, where $\mathrm{OPT} \in \mathcal{I}$ is any feasible subset of V'.

2. The Standard Greedy Algorithm

Before describing our general algorithm, let us recall the standard greedy algorithm, GREEDY, shown in Algorithm 1. The algorithm takes as input $\langle V, \mathcal{I}, f\rangle$, where V is a set of elements, $\mathcal{I} \subseteq 2^V$ is a hereditary constraint, represented as a membership oracle, and $f : 2^V \to \mathbb{R}_{\geq 0}$ is a

Algorithm 1 The standard greedy algorithm GREEDY

```
$S \leftarrow \emptyset$
loop
  Let $C = \{e \in V \setminus S : S \cup \{e\} \in \mathcal{I}\}$
  Let $e = \arg\max_{e \in C}\{f(S \cup \{e\}) - f(S)\}$
  if $C = \emptyset$ or $f(S \cup \{e\}) - f(S) < 0$ then
    return $S$
  end if
end loop
```

Algorithm 2 The distributed algorithm RANDGREEDI

```
for $e \in V$ do
  Assign $e$ to a machine $i$ chosen uniformly at random.
end for
Let $V_i$ be the elements assigned to machine $i$
Run GREEDY$(V_i)$ on each machine $i$ to obtain $S_i$
Place $S = \bigcup_i S_i$ on machine 1
Run ALG$(S)$ on machine 1 to obtain $T$
Let $S' = \arg\max_i\{f(S_i)\}$
return $\arg\max\{f(T), f(S')\}$
```

non-negative submodular function, represented as a value oracle. Given $\langle V, \mathcal{I}, f\rangle$, GREEDY iteratively constructs a solution $S \in \mathcal{I}$ by choosing at each step the element maximizing the marginal increase of f. For some $A \subseteq V$, we let GREEDY(A) denote the set $S \in \mathcal{I}$ produced by the greedy algorithm that considers only elements from A.

The greedy algorithm satisfies the following property:

Lemma 2. *Let $A \subseteq V$ and $B \subseteq V$ be two disjoint subsets of V. Suppose that, for each element $e \in B$, we have* GREEDY$(A \cup \{e\}) =$ GREEDY(A). *Then* GREEDY$(A \cup B) =$ GREEDY(A).

Proof. Suppose for contradiction that GREEDY$(A \cup B) \neq$ GREEDY(A). We first note that, if GREEDY$(A \cup B) \subseteq A$, then GREEDY$(A \cup B) =$ GREEDY(A); this follows from the fact that each iteration of the Greedy algorithm chooses the element with the highest marginal value whose addition to the current solution maintains feasibility for $\mathcal{I}$. Therefore, if GREEDY$(A \cup B) \neq$ GREEDY(A), the former solution contains an element of B. Let e be the first element of B that is selected by Greedy on the input $A \cup B$. Then Greedy will also select e on the input $A \cup \{e\}$, which contradicts the fact that GREEDY$(A \cup \{e\}) =$ GREEDY(A). ☐

3. A Randomized, Distributed Greedy Algorithm for Monotone Submodular Maximization

Algorithm. We now describe the specific variant of the GREEDI algorithm of Mirzasoleiman *et al.* that we consider. The algorithm, shown in Algorithm 2, proceeds exactly as GREEDI, except we perform the initial partitioning of V randomly.[4] Specifically, we suppose that each $e \in V$ is assigned to a machine chosen independently and uniformly at random. On each machine i, we execute GREEDY(V_i) to select a feasible subset S_i of the elements on that machine. In the second round, we place all of these selected subsets on a single machine, and run some algorithm ALG on this machine in order to select a final solution T. Finally, we return whichever is better: the final solution T or the best solution amongst all the S_i from the first phase. We call the resulting algorithm RANDGREEDI, to emphasize our assumption that the initial partitioning is performed randomly.

[4]Indeed, this was the case in several of the experiments performed by (Mirzasoleiman et al., 2013), and so our results provide some explanation for the gap between their worst-case bounds and experimental performance.

Analysis. We devote the rest of this section to the analysis of the RANDGREEDI algorithm. Fix $\langle V, \mathcal{I}, f\rangle$, where $\mathcal{I} \subseteq 2^V$ is a hereditary constraint, and $f : 2^V \to \mathbb{R}_{\geq 0}$ is any non-negative, monotone submodular function. Suppose that GREEDY is an α-approximation and ALG is a β-approximation for the associated constrained monotone submodular maximization problem of the form (1). Let $n = |V|$ and suppose that $\mathrm{OPT} = \arg\max_{A \in \mathcal{I}} f(A)$ is a feasible set maximizing f.

Let $\mathcal{V}(1/m)$ denote the distribution over random subsets of V where each element is included independently with probability $1/m$. Let $\mathbf{p} \in [0,1]^n$ be the following vector. For each element $e \in V$, we have

$$p_e = \begin{cases} \Pr_{A \sim \mathcal{V}(1/m)}[e \in \text{GREEDY}(A \cup \{e\})] & \text{if } e \in \mathrm{OPT} \\ 0 & \text{otherwise} \end{cases}$$

Our main theorem follows from the next two lemmas, which characterize the quality of the best solution from the first round and that of the solution from the second round, respectively. Recall that f^- is the Lovász extension of f.

Lemma 3. *For each machine i,* $\mathbb{E}[f(S_i)] \geq \alpha \cdot f^-(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p})$.

Proof. Consider machine i. Let V_i denote the set of elements assigned to machine i in the first round. Let $O_i = \{e \in \mathrm{OPT} \colon e \notin \text{GREEDY}(V_i \cup \{e\})\}$. We make the following key observations.

We apply Lemma 2 with $A = V_i$ and $B = O_i \setminus V_i$ to obtain that GREEDY$(V_i) =$ GREEDY$(V_i \cup O_i) = S_i$. Since $\mathrm{OPT} \in \mathcal{I}$ and $\mathcal{I}$ is hereditary, we must have $O_i \in \mathcal{I}$ as well. Since GREEDY is an α-approximation, it follows that

$$f(S_i) \geq \alpha \cdot f(O_i).$$

Since the distribution of V_i is the same as $\mathcal{V}(1/m)$, for each element $e \in \mathrm{OPT}$, we have

$$\Pr[e \in O_i] = 1 - \Pr[e \notin O_i] = 1 - p_e$$
$$\mathbb{E}[\mathbf{1}_{O_i}] = \mathbf{1}_{\mathrm{OPT}} - \mathbf{p}.$$

By combining these observations with Lemma 1, we obtain

$$\mathbb{E}[f(S_i)] \geq \alpha \cdot \mathbb{E}[f(O_i)] \geq \alpha \cdot f^{-}(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p}). \quad \square$$

Lemma 4. $\mathbb{E}[f(\textsc{Alg}(S))] \geq \beta \cdot f^{-}(\mathbf{p})$.

Proof. Recall that $S = \bigcup_i \textsc{Greedy}(V_i)$. Since $\mathrm{OPT} \in \mathcal{I}$ and $\mathcal{I}$ is hereditary, $S \cap \mathrm{OPT} \in \mathcal{I}$. Since $\textsc{Alg}$ is a β-approximation, we have

$$f(\textsc{Alg}(S)) \geq \beta \cdot f(S \cap \mathrm{OPT}). \tag{2}$$

Consider an element $e \in \mathrm{OPT}$. For each machine i, we have

$$\begin{aligned}
&\Pr[e \in S \mid e \text{ is assigned to machine } i] \\
&\quad = \Pr[e \in \textsc{Greedy}(V_i) \mid e \in V_i] \\
&\quad = \Pr_{A \sim \mathcal{V}(1/m)}[e \in \textsc{Greedy}(A) \mid e \in A] \\
&\quad = \Pr_{B \sim \mathcal{V}(1/m)}[e \in \textsc{Greedy}(B \cup \{e\})] \\
&\quad = p_e.
\end{aligned}$$

The first equality follows from the fact that e is included in S if and only if it is included in $\textsc{Greedy}(V_i)$. The second equality follows from the fact that the distribution of V_i is identical to $\mathcal{V}(1/m)$. The third equality follows from the fact that the distribution of $A \sim \mathcal{V}(1/m)$ conditioned on $e \in A$ is identical to the distribution of $B \cup \{e\}$ where $B \sim \mathcal{V}(1/m)$. Therefore, $\Pr[e \in S \cap \mathrm{OPT}] = p_e$ and so $\mathbb{E}[\mathbf{1}_{S \cap \mathrm{OPT}}] = \mathbf{p}$. Lemma 1 thus implies that

$$\mathbb{E}[f(\textsc{Alg}(S))] \geq \beta \cdot \mathbb{E}[f(S \cap \mathrm{OPT})] \geq \beta \cdot f^{-}(\mathbf{p}). \quad \square$$

Combining Lemma 4 and Lemma 3 gives us our main theorem.

Theorem 5. *Suppose that* $\textsc{Greedy}$ *is an* α*-approximation algorithm and* $\textsc{Alg}$ *is a* β*-approximation algorithm for maximizing a monotone submodular function subject to a hereditary constraint* $\mathcal{I}$*. Then* $\textsc{RandGreeDi}$ *is (in expectation) an* $\frac{\alpha\beta}{\alpha+\beta}$*-approximation algorithm for the same problem.*

Proof. Let $S_i = \textsc{Greedy}(V_i)$, $S = \bigcup_i S_i$ be the set of elements on the last machine, and $T = \textsc{Alg}(S)$ be the solution produced on the last machine. Then, the output D of $\textsc{RandGreeDi}$ satisfies $f(D) \geq \max_i \{f(S_i)\}$ and $f(D) \geq f(T)$. Thus, from Lemmas 3 and 4 we have:

$$\mathbb{E}[f(D)] \geq \alpha \cdot f^{-}(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p}) \tag{3}$$

$$\mathbb{E}[f(D)] \geq \beta \cdot f^{-}(\mathbf{p}). \tag{4}$$

By combining (3) and (4), we obtain

$$\begin{aligned}
(\beta + \alpha)\,\mathbb{E}[f(D)] &\geq \alpha\beta\big(f^{-}(\mathbf{p}) + f^{-}(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p})\big) \\
&\geq \alpha\beta \cdot f^{-}(\mathbf{1}_{\mathrm{OPT}}) = \alpha\beta \cdot f(\mathrm{OPT}).
\end{aligned}$$

In the second inequality, we have used the fact that f^{-} is convex and $f^{-}(c \cdot \mathbf{x}) \geq c f^{-}(\mathbf{x})$ for any $c \in [0, 1]$. $\square$

If we use the standard $\textsc{Greedy}$ algorithm for $\textsc{Alg}$, we obtain the following simplified corollary of Theorem 5.

Corollary 6. *Suppose that* $\textsc{Greedy}$ *is an* α*-approximation algorithm for maximizing a monotone submodular function, and use* $\textsc{Greedy}$ *as the algorithm* $\textsc{Alg}$ *in* $\textsc{RandGreeDi}$*. Then, the resulting algorithm is (in expectation) an* $\frac{\alpha}{2}$*-approximation algorithm for the same problem.*

4. Non-Monotone Submodular Functions

We consider the problem of maximizing a *non-monotone* submodular function subject to a hereditary constraint. Our approach is a slight modification of the randomized, distributed greedy algorithm described in Section 3, and it builds on the work of (Gupta et al., 2010). Again, we show how to combine the standard $\textsc{Greedy}$ algorithm, together with any algorithm $\textsc{Alg}$ for the non-monotone case in order to obtain a randomized, distributed algorithm for non-monotone submodular maximization.

Algorithm. Our modified algorithm, $\textsc{NMRandGreeDi}$, works as follows. As in the monotone case, in the first round we distribute the elements of V uniformly at random amongst the m machines. Then, we run the standard greedy algorithm *twice* to obtain two disjoint solutions S_i^1 and S_i^2 on each machine. Specifically, each machine first runs $\textsc{Greedy}$ on V_i to obtain a solution S_i^1, then runs $\textsc{Greedy}$ on $V_i \setminus S_i^1$ to obtain a disjoint solution S_i^2. In the second round, both of these solutions are sent to a single machine, which runs $\textsc{Alg}$ on $S = \bigcup_i (S_i^1 \cup S_i^2)$ to produce a solution T. The best solution amongst T and all of the solutions S_i^1 and S_i^2 is then returned.

Analysis. We devote the rest of this section to the analysis of the algorithm. In the following, we assume that we are working with an instance $\langle V, \mathcal{I}, f \rangle$ of non-negative, non-monotone submodular maximization for which the $\textsc{Greedy}$ algorithm satisfies the following property (for some γ):

$$\text{For all } S \in \mathcal{I}\text{: } f(\textsc{Greedy}(V)) \geq \gamma \cdot f(\textsc{Greedy}(V) \cup S) \tag{GP}$$

The standard analyses of the $\textsc{Greedy}$ algorithm show that (GP) is satisfied with $\gamma = \frac{1}{2}$ for cardinality and matroid constraints, $\gamma = \frac{1}{3}$ for knapsack constraints, and $\gamma = \frac{1}{p+1}$ for p-system constraints.

The analysis is similar to the approach from the previous section. We define $\mathcal{V}(1/m)$ as before, but modify the definition of the vector $\mathbf{p}$ as follows: for each $e \in V \setminus \mathrm{OPT}$ we let $p_e = 0$ and for each $e \in \mathrm{OPT}$, we let p_e be:

$$\Pr_{A \sim \mathcal{V}(1/m)}\big[e \in \textsc{Greedy}(A \cup \{e\}) \textbf{ or } e \in \textsc{Greedy}((A \cup \{e\}) \setminus \textsc{Greedy}(A \cup \{e\}))\big].$$

We now give analogues of Lemmas 3 and 4. The proof of the Lemma 8 is similar to that of Lemma 4, and is deferred to the supplement.

Lemma 7. *Suppose that* Greedy *satisfies* (GP). *For each machine i,* $\mathbb{E}\left[\max\{f(S_i^1), f(S_i^2)\}\right] \geq \frac{\gamma}{2} \cdot f^-(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p})$.

Proof. Consider machine i and let V_i be the set of elements assigned to machine i in the first round. Let

$$O_i = \{e \in \mathrm{OPT} \colon e \notin \textsc{Greedy}(V_i \cup \{e\}) \textbf{ and } e \notin \textsc{Greedy}((V_i \cup \{e\}) \setminus \textsc{Greedy}(V_i \cup \{e\}))\}$$

Note that, since $\mathrm{OPT} \in \mathcal{I}$ and $\mathcal{I}$ is hereditary, we have $O_i \in \mathcal{I}$. It follows from Lemma 2 that

$$S_i^1 = \textsc{Greedy}(V_i) = \textsc{Greedy}(V_i \cup O_i),$$
$$S_i^2 = \textsc{Greedy}(V_i \setminus S_i^1) = \textsc{Greedy}((V_i \setminus S_i^1) \cup O_i).$$

By combining the equations above with the greedy property (GP), we obtain

$$f(S_i^1) \geq \gamma \cdot f(S_i^1 \cup O_i), \tag{5}$$
$$f(S_i^2) \geq \gamma \cdot f(S_i^2 \cup O_i). \tag{6}$$

Now we observe that the submodularity and non-negativity of f, together with $S_i^1 \cap S_i^2 = \emptyset$, imply

$$f(S_i^1 \cup O_i) + f(S_i^2 \cup O_i) \geq f(O_i). \tag{7}$$

By combining (5), (6), and (7), we obtain

$$f(S_i^1) + f(S_i^2) \geq \gamma \cdot f(O_i). \tag{8}$$

Since the distribution of V_i is the same as $\mathcal{V}(1/m)$, for each element $e \in \mathrm{OPT}$, we have

$$\Pr[e \in O_i] = 1 - \Pr[e \notin O_i] = 1 - p_e,$$
$$\mathbb{E}[\mathbf{1}_{O_i}] = \mathbf{1}_{\mathrm{OPT}} - \mathbf{p}. \tag{9}$$

By combining (8), (9), and Lemma 1, we obtain

$$\mathbb{E}[f(S_i^1) + f(S_i^2)] \geq \gamma \cdot \mathbb{E}[f(O_i)] \geq \gamma \cdot f^-(\mathbf{1}_{\mathrm{OPT}} - \mathbf{p}),$$

which immediately implies the desired inequality. □

Lemma 8. $\mathbb{E}[f(\textsc{Alg}(S))] \geq \beta \cdot f^-(\mathbf{p})$.

Lemmas 7 and 8 imply our main result for non-monotone submodular maximization (the proof is similar to that of Theorem 5).

Theorem 9. *Consider the problem of maximizing a submodular function under some hereditary constraint $\mathcal{I}$, and suppose that* Greedy *satisfies* (GP) *and* Alg *is a β-approximation algorithm for this problem. Then* NMRandGreeDi *is (in expectation) an $\frac{\gamma\beta}{\gamma+2\beta}$-approximation algorithm for the same problem.*

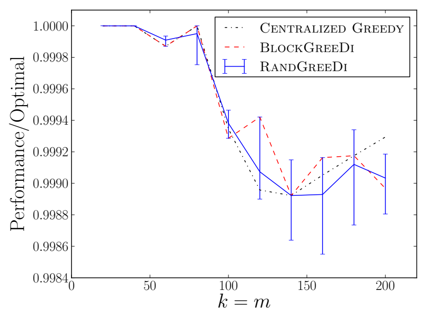
We remark that one can use the following approach on the last machine (Gupta et al., 2010). As in the first round, we run Greedy twice to obtain two solutions $T_1 = \textsc{Greedy}(S)$ and $T_2 = \textsc{Greedy}(S \setminus T_1)$. Additionally, we select a subset $T_3 \subseteq T_1$ using an *unconstrained* submodular maximization algorithm on T_1, such as the Double Greedy algorithm of (Buchbinder et al., 2012), which is a $\frac{1}{2}$-approximation. The final solution T is the best solution among T_1, T_2, T_3. If Greedy satisfies property (GP), then it follows from the analysis of (Gupta et al., 2010) that the resulting solution T satisfies $f(T) \geq \frac{\gamma}{2(1+\gamma)} \cdot f(\mathrm{OPT})$. This gives us the following corollary of Theorem 9.

Corollary 10. *Consider the problem of maximizing a submodular function subject to some hereditary constraint $\mathcal{I}$ and suppose that* Greedy *satisfies* (GP) *for this problem. Let* Alg *be the algorithm described above. Then* NMRandGreeDi *achieves (in expectation) an $\frac{\gamma}{4+2\gamma}$-approximation for the same problem.*

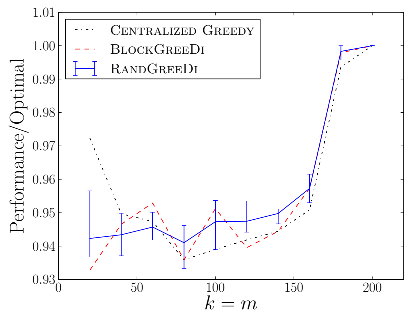
5. Experiments

We experimentally evaluate and compare the following distributed algorithms for maximizing a monotone submodular function subject to a cardinality constraint: the randomized variant of the GreeDi algorithm described in Sections 3 and 4, a deterministic variant of the GreeDi algorithm that assigns elements to machines in consecutive blocks of size $|V|/m$, and the Sample&Prune algorithm of (Kumar et al., 2013). We run these algorithms in several scenarios and we evaluate their performance relative to the centralized Greedy solution on the entire dataset.

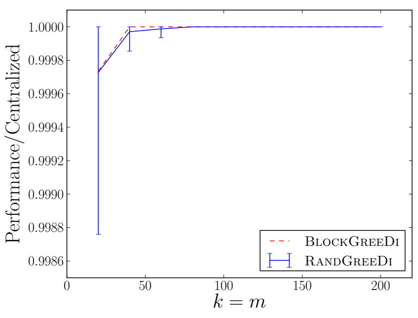
Exemplar based clustering. Our experimental setup is similar to that of (Mirzasoleiman et al., 2013). Our goal is to find a representative set of objects from a dataset by solving a k-medoid problem (Kaufman & Rousseeuw, 2009) that aims to minimize the sum of pairwise dissimilarities between the chosen objects and the entire dataset. Let V denote the set of objects in the dataset and let $d : V \times V \to \mathbb{R}$ be a dissimilarity function; we assume that d is symmetric, that is, $d(i,j) = d(j,i)$ for each pair i, j. Let $L : 2^V \to \mathbb{R}$ be the function such that $L(A) = \frac{1}{|V|}\sum_{v \in V} \min_{a \in A} d(a, v)$ for each set $A \subseteq V$. We can turn the problem of minimizing L into the prob-



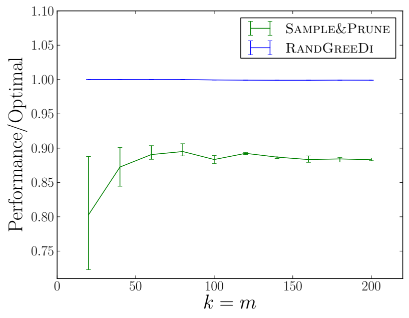
(a) Kosarak dataset



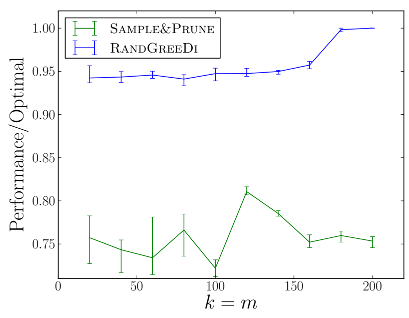
(b) accidents dataset



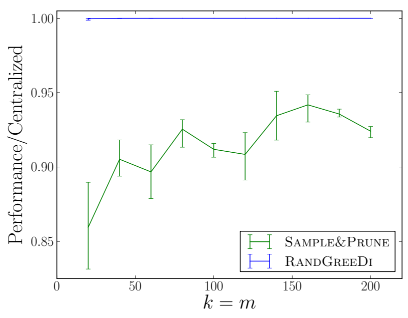
(c) 10K tiny images



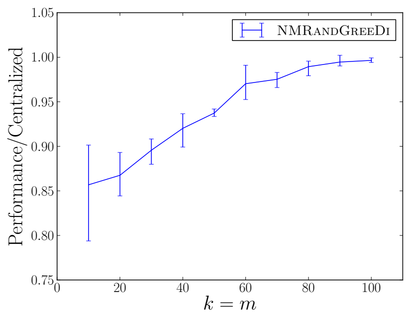
(d) Kosarak dataset



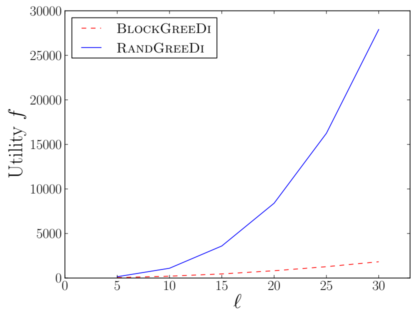
(e) accidents dataset



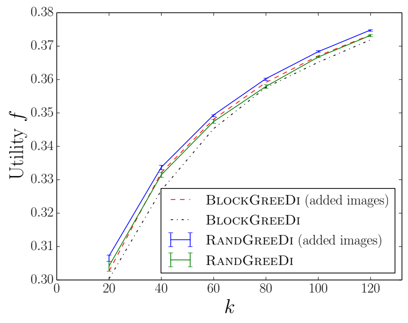
(f) 10K tiny images



(g) synthetic diverse-yet-relevant instance ($n = 10000$, $\lambda = n/k$)



(h) synthetic hard instance for GreeDi



(i) 1M tiny images

Figure 1. Experiment Results (I)

lem of maximizing a monotone submodular function f by introducing an auxiliary element v_0 and by defining $f(S) = L(\{v_0\}) - L(S \cup \{v_0\})$ for each set $S \subseteq V$.

Tiny Images experiments: In our experiments, we used a subset of the Tiny Images dataset consisting of 32×32 RGB images (Torralba et al., 2008), each represented as $3,072$ dimensional vector. We subtracted from each vector the mean value and normalized the result, to obtain a collection of $3,072$-dimensional vectors of unit norm. We considered the distance function $d(x, y) = \|x - y\|^2$ for every pair x, y of vectors. We used the zero vector as the auxiliary element v_0 in the definition of f.

In our smaller experiments, we used 10,000 tiny images, and compared the utility of each algorithm to that of the centralized Greedy. The results are summarized in Figures 1(c) and 1(f).

In our *large scale experiments*, we used one million tiny images, and $m = 100$ machines. In the first round of the distributed algorithm, each machine ran the Greedy algorithm to maximize a restricted objective function f, which is based on the average dissimilarity L taken over only those images assigned to that machine. Similarly, in the second round, the final machine maximized an objective function f based on the total dissimilarity of all those images it received . We also considered a variant similar to that described by (Mirzasoleiman et al., 2013), in which 10,000 additional random images from the original dataset were added to the final machine. The results are summarized in Figure 1(i).

Remark on the function evaluation. In decomposable cases such as exemplar clustering, the function is a sum of distances over all points in the dataset. By concentration results such as Chernoff bounds, the sum can be approximated additively with high probability by sampling a few points and using the (scaled) empirical sum. The random subset each machine receives can readily serve as the samples for the above approximation. Thus the random partition is useful for evaluating the function in a distributed fashion, in addition to its algorithmic benefits.

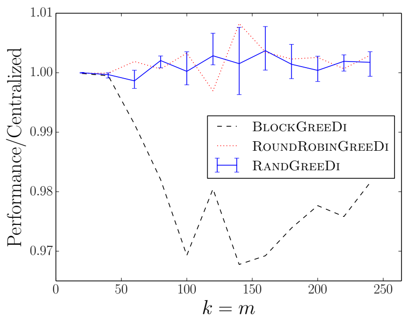
Maximum Coverage experiments. We ran several experiments using instances of the Maximum Coverage problem. In the Maximum Coverage problem, we are given a collection $\mathcal{C} \subseteq 2^V$ of subsets of a ground set V and an integer k, and the goal is to select k of the subsets in $\mathcal{C}$ that cover as many elements as possible.

Kosarak and accidents datasets: We evaluated and compared the algorithms on the datasets used in (Kumar et al., 2013). In both cases, we computed the optimal centralized solution using CPLEX, and calculated the actual performance ratio attained by the algorithms. The results are summarized in Figures 1(a), 1(d), 1(b), 1(e).

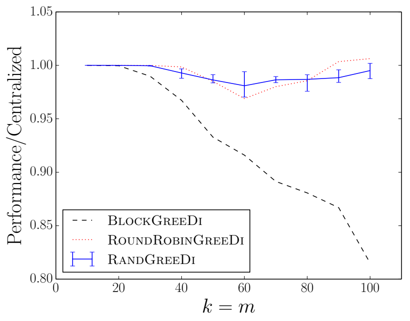
Synthetic hard instances: We generated a synthetic dataset with hard instances for the deterministic GREEDI. We describe the instances in the supplement. We ran the GREEDI algorithm with a worst-case partition of the data. The results are summarized in Figure 1(h).

Finding diverse yet relevant items. We evaluated the randomized algorithm NMRANDGREEDI described in Section 4 on the following instance of *non-monotone* submodular maximization subject to a cardinality constraint. We used the objective function of (Lin & Bilmes, 2009): $f(A) = \sum_{i \in V} \sum_{j \in A} s_{ij} - \lambda \sum_{i,j \in A} s_{ij}$, where λ is a redundancy parameter and $\{s_{ij}\}_{ij}$ is a similarity matrix. We generated an $n \times n$ similarity matrix with random entries $s_{ij} \in \mathcal{U}(0, 100)$ and we set $\lambda = n/k$. The results are summarized in Figure 1(g).

Matroid constraints. In order to evaluate our algorithm on a matroid constraint, we considered the following variant of maximum coverage: we are given a space containing several demand points and n facilities (e.g. wireless access points or sensors). Each facility can operate in one of r modes, each with a distinct coverage profile. The goal is to find a subset of at most k facilities to activate, along with a single mode for each activated facility, so that the total number of demand points covered is maximized. In our experiment, we placed 250,000 demand points in a grid in the unit square, together with a grid of n facilities. We modeled coverage profiles as ellipses centered at each facility with major axes of length 0.1ℓ, minor axes of length $0.1/\ell$ rotated by ρ where $\ell \in \mathcal{N}(3, \frac{1}{3})$ and $\rho \in \mathcal{U}(0, 2\pi)$ are chosen randomly for each ellipse. We performed two series of experiments. In the first, there were $n = 900$ facilities, each with $r = 5$ coverage profiles, while in the second there were $n = 100$ facilities, each with $r = 100$ coverage profiles.



(a) matroid coverage ($n = 900, r = 5$)



(b) matroid coverage ($n = 100, r = 100$)

Figure 2. Experiment Results (II)

The resulting problem instances were represented as ground set comprising a list of ellipses, each with a designated facility, together with a partition matroid constraint ensuring that at most one ellipse per facility was chosen. Here, we compared the randomized GREEDI algorithm to two deterministic variants that assigned elements to machines in consecutive blocks and in round robin order, respectively. The results are summarized in Figures 2(a) and 2(b).

Acknowledgements. We thank Moran Feldman for suggesting a modification to our original analysis that led to the simpler and stronger analysis included in this version of the paper. This work was supported by EPSRC grant EP/J021814/1.

References

Buchbinder, Niv, Feldman, Moran, Naor, Joseph, and Schwartz, Roy. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *Foundations of Computer Science (FOCS)*, pp. 649–658. IEEE, 2012.

Dean, Jeffrey and Ghemawat, Sanjay. Mapreduce: Simplified data processing on large clusters. In *Symposium on Operating System Design and Implementation (OSDI)*, pp. 137–150. USENIX Association, 2004.

Fisher, Marshall L, Nemhauser, George L, and Wolsey, Laurence A. An analysis of approximations for maximizing submodular set functions—II. *Mathematical Programming Studies*, 8:73–87, 1978.

Gupta, Anupam, Roth, Aaron, Schoenebeck, Grant, and Talwar, Kunal. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *Internet and Network Economics*, pp. 246–257. Springer, 2010.

Indyk, Piotr, Mahabadi, Sepideh, Mahdian, Mohammad, and Mirrokni, Vahab S. Composable core-sets for diversity and coverage maximization. In *ACM Symposium on Principles of Database Systems (PODS)*, pp. 100–108. ACM, 2014.

Jenkyns, Thomas A. The efficacy of the "greedy" algorithm. In *Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pp. 341–350. Utilitas Mathematica, 1976.

Kaufman, Leonard and Rousseeuw, Peter J. *Finding groups in data: An introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

Kumar, Ravi, Moseley, Benjamin, Vassilvitskii, Sergei, and Vattani, Andrea. Fast greedy algorithms in mapreduce and streaming. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pp. 1–10. ACM, 2013.

Lin, Hui and Bilmes, Jeff A. How to select a good training-data subset for transcription: Submodular active selection for sequences. In *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Brighton, UK, September 2009.

Mirzasoleiman, Baharan, Karbasi, Amin, Sarkar, Rik, and Krause, Andreas. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2049–2057, 2013.

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

Torralba, Antonio, Fergus, Robert, and Freeman, William T. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008.

Wolsey, Laurence A. Maximising real-valued submodular functions: Primal and dual heuristics for location problems. *Mathematics of Operations Research*, 7(3): pp. 410–425, 1982.