Fast Constrained Submodular Maximization: Personalized Data Summarization

Baharan Mirzasoleiman BAHARANM@INF.ETHZ.CH
ETH Zurich

Ashwinkumar Badanidiyuru ASHWINKUMARBV@GOOGLE.COM
Google

Amin Karbasi AMIN.KARBASI@YALE.EDU
Yale University

Abstract

Can we summarize multi-category data based on user preferences in a scalable manner? Many utility functions used for data summarization satisfy submodularity, a natural diminishing returns property. We cast personalized data summarization as an instance of a general submodular maximization problem subject to multiple constraints. We develop the first practical and FAst coNsTrained submOdular Maximization algorithm, FANTOM, with strong theoretical guarantees. FANTOM maximizes a submodular function (*not necessarily monotone*) subject to the intersection of a p-system and l knapsacks constrains. It achieves a $(1+\epsilon)(p+1)(2p+2l+1)/p$ approximation guarantee with only $O(\frac{nrp\log(n)}{\epsilon})$ query complexity (n and r indicate the size of the ground set and the size of the largest feasible solution, respectively). We then show how we can use FANTOM for personalized data summarization. In particular, a p-system can model different aspects of data, such as categories or time stamps, from which the users choose. In addition, knapsacks encode users' constraints including budget or time. In our set of experiments, we consider several concrete applications: movie recommendation over $11K$ movies, personalized image summarization with $10K$ images, and revenue maximization on the YouTube social networks with 5000 communities. We observe that FANTOM constantly provides the highest utility against all the baselines.

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

1. Introduction

A tremendous amount of data is generated every second, every day by tens of social media and millions of users empowering them. Recent statistics indicate that every minute Instagram users post nearly 220,000 new photos, YouTube users upload 72 hours of video, and Facebook users share nearly 2.5 million pieces of content. Organizing and making sense of big data has become one of today's major challenges in machine learning. Data summarization, in the form of extracting a representative subset of data points, is a natural way to obtain a faithful description of the whole data. In general, a representative summary has two requirements [Tschiatschek et al., 2014; Dasgupta et al., 2013]:

- **Coverage:** A good summary is concise so that it contains elements from distinct parts of data. Naturally, a concise summary minimizes information loss.
- **Diversity:** A good summary is compact so that it does not contain elements that are too similar to each other.

Very often, the utility/scoring function f capturing the coverage or diversity of a subset w.r.t. the entire dataset E satisfies submodularity, an intuitive diminishing returns condition [Lin and Bilmes, 2011; Mirzasoleiman et al., 2013]. In words, adding a new element to a smaller summary, adds more value than adding the same element to its superset.

Note that coverage and diversity could sometimes be conflicting requirements: higher coverage usually means selecting more elements whereas higher diversity penalizes having similar elements in the summary and prevents the summary from growing too large. Depending on the application, a good summary can trade off between coverage and diversity by putting more emphasis on one or the other. By design, utility functions expressing coverage are *monotone* as it is quit natural to assume that

adding more elements to a summary will only decrease the information loss. Such monotone submodular functions have been extensively used for many data summarization applications including clustering [Dueck and Frey, 2007; Gomes and Krause, 2010], scene summarization [Simon et al., 2007], document and corpus summarization [Lin and Bilmes, 2011; Sipos et al., 2012], recommender systems [El-Arini and Guestrin, 2011], crowd teaching [Singla et al., 2014], and active set selection in kernel machines [Scholkopf and Smola, 2001; Mirzasoleiman et al., 2013]. In contrast, utility functions that accommodate diversity are not necessarily monotone as they penalize larger solutions [Tschiatschek et al., 2014; Dasgupta et al., 2013]. Consequently, the functions designed to measure both coverage and diversity (e.g., combination of monotone submodular functions and decreasing penalty terms) are naturally *non-monotone*.

On top of maximizing a non-monotone submodular utility function, there are often constraints imposed by the underlying data summarization application. For instance, an individual interested in showing a summary of her recent trip photos may not intend to include more than a handful of them (i.e., cardinality constraint). Or, a user interested in watching representative video clips (with different duration) from a particular category may not wish to spend more than a certain amount of time (i.e., knapsack constraint). In general, we can cast the data summarization problem as that of selecting a subset of elements S from a large dataset E while optimizing a submodular utility function f that quantifies the representativeness of the selected subset subject to feasibility constraints ξ:

$$\max_{S \subseteq E} f(S) \text{ s.t. } S \in \xi. \tag{1}$$

There exist fast and scalable methods to solve problem (1) for a *monotone* submodular function f with a variety of constraints [Badanidiyuru and Vondrák, 2014; Mirzasoleiman et al., 2015; Badanidiyuru et al., 2014; Mirzasoleiman et al., 2013; Wei et al., 2014; Kumar et al., 2015]. As a result, monotone submodular maximization subject to simple constraints (often a cardinality constraint) has been one of the prototypical optimization problems for data summarization. In this paper, we aim to significantly push the theoretical boundaries of constrained submodular maximization while providing a practical data summarization method for far richer scenarios. We develop a FAst coNsTrained submOdular Maximization algorithm, FANTOM, for solving the optimization problem (1) where f is not necessarily monotone and ξ can be as general as the intersection of a p-system and l knapsacks. We show that FANTOM provides a solution with $(1+\epsilon)(p+1)(2p+2l+1)/p$ approximation guarantee with $O(\frac{nrp\log(n)}{\epsilon})$ query complexity (n and r indicate the size of the ground set and the size of the largest feasible solution, respectively). To the best of our knowledge, there is no algorithm with such strong guarantees for maximizing a *general* submodular function under the aforementioned constrains. Moreover, even in the case of a single matroid and a single knapsack constraint, the best known algorithms suffer from a prohibitive running time that makes them impractical for any effective data summarization applications (see Section 2). Last but not least, a p-system contains cardinality, matroid, and the intersection of p matroids, as special cases. Thus, it allows us to easily model various data summarization scenarios for which only heuristic methods were known. We discuss the personalized data summarization in Section 4 with three concrete applications: movie recommendation on a dataset containing $11K$ movies, personalized image summarization on a multi-category dataset with $10K$ images, and revenue maximization on the YouTube social network with 5000 communities.

2. Related Work

There has been a recent surge of interest in applying submodular optimization methods to machine learning applications, including viral marketing [Kempe et al., 2003], network monitoring [Leskovec et al., 2007], sensor placement and information gathering [Krause and Guestrin, 2011], news article recommendation [El-Arini et al., 2009], nonparametric learning [Gomes and Krause, 2010; Reed and Ghahramani, 2013], document and corpus summarization [Lin and Bilmes, 2011; Dasgupta et al., 2013; Sipos et al., 2012], crowd teaching [Singla et al., 2014], and MAP inference of determinental point process [Gillenwater et al., 2012]. A key reason for such a wide range of applications is the existence of efficient algorithms (with near-optimal solutions) for a diverse set of constraints when the submodular function is monotone.

Constrained monotone submodular maximization: The literature on constrained maximization of monotone submodular functions is very rich. For this problem, we know a wide variety of algorithms such as the well-known greedy [Nemhauser et al., 1978], the continuous greedy [Vondrák, 2008], and the local search algorithm [Lee et al., 2009]. We also know algorithms for this problem with good approximation ratios under a wide variety of constraints such as p-system, l-knapsacks [Badanidiyuru and Vondrák, 2014], and the orienteering problem [Chekuri and Pál, 2005]. Additionally, very efficient centralized algorithms [Badanidiyuru and Vondrák, 2014; Mirzasoleiman et al., 2015], scalable algorithms in streaming [Badanidiyuru et al., 2014; Chekuri et al., 2015a; Chakrabarti and Kale, 2015], and distributed [Mirzasoleiman et al., 2013; Kumar et al., 2015] settings have also been developed.

Constrained non-monotone submodular maximization: While maximizing monotone submodular functions has been applied to many machine learning applications, the

Constraint	Previous		New	
	Approximation	Running time	Approximation	Running time
p-system + l-knapsack	-	-	$(1+\epsilon)(p+1)(2p+2\ell+1)/p$	$O(\frac{nrp\log(n)}{\epsilon})$
1-matroid + l-knapsack	$e+\epsilon$ [Feldman et al., 2011] [Chekuri et al., 2014]	poly$(n)\cdot$ exp(l,ϵ)		
p-matroid + l-knapsack	$p/0.19+\epsilon$ [Chekuri et al., 2014]	poly$(n)\cdot$ exp(p,l,ϵ)		
p-system	$(p+1)(3p+3)/p$ [Gupta et al., 2010]	$O(nrp)$	$(p+1)(2p+1)/p$	$O(nrp)$
p-matroid	$p+1+1/(p-1)+\epsilon$ [Lee et al., 2010]	poly$(n)\cdot$ exp(p,ϵ)		

Table 1. Comparison of running times and approximation ratios for non-monotone submodular maximization under different constraints.

problem of maximizing non-monotone submodular functions has not found as many applications. Part of the reason is that the existing algorithms for handling generic constraints such as both matroid and knapsack constraints are very slow. A body of work [Feldman et al., 2011; Chekuri et al., 2014; Gupta et al., 2010; Lee et al., 2010; Feige et al., 2011; Buchbinder et al., 2015] has found algorithms with good approximation ratios, albeit with running times of very high polynomial. A comparison can be found in Table 1. In particular, for l-matroid and l-knapsack constraints, a $(e+\epsilon)$-approximation follows from applying results from [Feldman et al., 2011] to contention resolution techniques from [Chekuri et al., 2014]. Feldman et al. don't formalize this as a theorem but state it in the introduction. Gupta et al. [2010] and Gupta et al. [2015] further showed a $O(p)$ approximation for p-matroid and l-knapsack constraints. This result is summarized in Table 1 of [Chekuri et al., 2014] as a $(p/0.19+\epsilon)$- approximation. For a p-system constraint without any knapsack constraint, the approximation ratio of $(p+1)(3p+3)/p$ can be obtained by substituting $\alpha=1/2$ approximation for unconstrained maximization into Theorem 3.3 of [Gupta et al., 2010]. Finally, p-matroid approximation ratio of $p+1+1/(p-1)+\epsilon$ is provided by Theorem 4 of [Lee et al., 2010].

It is worth mentioning that in addition to the above results, recently Chekuri et al. [Chekuri et al., 2015b] developed a continuous-time framework that provides a $(1-1/e-\epsilon)$ approximation for maximizing a submodular function under packing constraints. Although their algorithm gives a $O(n^2)$ runtime for the fractional solution to the multi-linear extension, converting the fractional solution to an intergral solution requires enumerating sets of size poly$(1/\epsilon)$. This results in a run time of $n^{\text{poly}(1/\epsilon)}$, which is impractical for most real-world scenarios. In this paper, we develop the first practical algorithm, FANTOM, for maximizing non-monotone submodular functions with very generic p-system and l-knapsack constraints.

3. Problem Statement

Let E be the ground set of elements. A set function $f: 2^E \to \mathbb{R}_+$ is *submodular* if for any two sets $S \subseteq T \subseteq E$ and any element $e \in E \setminus T$ we have that

$$f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T).$$

It is monotone if $f(S \cup \{e\}) - f(S) \geq 0$ for all $S \subseteq E$ and $e \in E \setminus S$. A *matroid* is a pair $M = (E, \mathcal{I})$, where $\mathcal{I}$ is a family of subsets of E (called *independent* sets) with the following three properties: 1) $\emptyset \in \mathcal{I}$, 2) for each $A \subseteq B \subseteq E$, if $B \in \mathcal{I}$ then $A \in \mathcal{I}$, and 3) for every $A, B \in \mathcal{I}$ if $|A| < |B|$, then there exists an $e \in B \setminus A$, such that $A \cup \{e\} \in \mathcal{I}$. For a matroid, the size of all maximal independent sets are equal (called *rank*). Two common matroids are the uniform, and partition matroids. A *uniform* matroid is the family of all subsets of size at most k. In a *partition* matroid, we have a collection of disjoint sets B_i and integers $0 \leq d_i \leq |B_i|$ where a set A is independent if for every index i, we have $|A \cap B_i| \leq d_i$. A p-system is a pair $M = (E, \mathcal{I})$ such that 1) $\emptyset \in \mathcal{I}$, 2) for each $A \subseteq B \subseteq E$, if $B \in \mathcal{I}$ then $A \in \mathcal{I}$, and 3) if $A, B \in \mathcal{I}$ are two maximal sets, then $|A| \leq p|B|$. It is useful to note that the intersection of p matroids forms a p-system. A *knapsack* constraint is defined by a cost function $c: E \to \mathbb{R}_+$. A set $S \subseteq E$ is said to satisfy the knapsack constraint if $c(S) = \sum_{e \in S} c(e) \leq 1$.

Our goal in this paper is to maximize a (non-monotone) submodular function f subject to a set of constraints ξ defined by the intersection of a p-system $(E, \mathcal{I})$ and l knapsacks. In other words, we would like to find a set $S \in \mathcal{I}$ that maximizes f where for each knapsack c_i (where $1 \leq i \leq l$) we have $\sum_{e \in S} c_i(e) \leq 1$. For the ease of presentation, we use c_{ij} to denote the cost of element $j \in E$ in the i-th knapsack. Before explaining how we solve this problem, we discuss three concrete applications for which we need to find fast solutions.

4. Personalized Data Summarization

In this part, we discuss three concrete applications with their corresponding utility functions and constraints ξ.

Personalized recommendation: Consider a movie recommender system, where a user specifies the genres she is interested in, out of l categories, and the recommender system has to provide a short list of representative movies accordingly. To this end, we represent each movie by a vector consist of users' ratings. Such a representation can be easily obtained by using existing low-rank matrix completion techniques [Candès and Recht, 2009] that provide a complete rating matrix $M_{k\times n}$ based on few ratings of k users for n movies in the ground set E. By forming M, we can measure the similarity $s_{i,j}$ between movies i and j through the inner product between the corresponding columns. Note that a movie can be a member of different categories (e.g., a movie can be both drama and comedy). We denote by $G(i)$ the genres of movie $i \in E$. We also let E_g denote the set of movies from genre g. Clearly, two genres g, g' may overlap, i.e., $E_g \cap E_{g'} \neq \emptyset$. Moreover, each item has a cost that can represent the monetary cost, duration, or even accessibility of the movie, among many other factors. The recommender system has to provide a *short list* that meets the *user's constraints*, in terms of money, time, or accessibility. To model this scenario, we use intersection of l uniform matroids to prevent each category from having more than a certain number of movies. A knapsack constraint is also used to model the user's limitation in terms of the money she can pay, the time she can spend, or how much effort she has to make to find such movies. A sensible submodular utility function that we can use in order to score the quality of the selected movies is

$$f(S) = \sum_{i\in E}\sum_{j\in S} s_{i,j} - \lambda \sum_{i\in S}\sum_{j\in S} s_{i,j}, \qquad (2)$$

for some $0 \leq \lambda \leq 1$. Note that for $\lambda = 1$ the above function is the cut-function. This utility function is non-negative and non-monotone. The first term is the sum-coverage function (to capture coverage) and the second term penalizes similarity within S (to capture diversity). Such functions have been previously used in document [Lin and Bilmes, 2011] and scene summarization [Simon et al., 2007]. Another possible way to compute the similarity between a movie i and the dataset E is to consider only movies which have a common genre with i as follows

$$f(S) = \sum_{j\in S}\sum_{g\in G(j)}\sum_{i\in E_g} s_{i,j} - \lambda \sum_{j\in S}\sum_{g\in G(j)}\sum_{i\in E_g\cap S} s_{i,j}. \qquad (3)$$

This way the utility function emphasizes on movies that have more common genres with what the user desires.

Personalized image summarization: Here, we have a collection of images E from l disjoint categories (e.g., mountains, bikes, birthdays, etc) and the user is interested in a summary only from a few categories. For simplicity, we assume that each image is only a member of a single category. This assumption lets us define a partition matroid consisting of l groups. The user basically identifies the set of desired groups. The size of groups puts a limit on the number of images that can be chosen from each group. The cost of an image is chosen as a function of its quality, such as the resolution, contrast, luminance, etc. For the utility function, we can use

$$f(S) = \sum_{i\in E}\max_{j\in S} d_{i,j} - \frac{1}{|E|}\sum_{i\in S}\sum_{j\in S} d_{i,j}, \qquad (4)$$

where $d_{i,j}$ determines the similarity of image i to image j. There are many ways to determine the similarity between images such as cosine similarity or a distance metric. The first term is the facility location objective function (for coverage) and the second term is a dispersion function (for diversity). Facility location has been extensively used for image summarization in the form of exemplar-based clustering [Dueck and Frey, 2007; Gomes and Krause, 2010]. The above submodular function is non-negative and non-monotone.

Revenue maximization with multiple products: In this application, we consider revenue maximization on a social network $G = (V, W)$ when multiple products from a basket Q that can be offered to each user $i \in V$. Here, we assume that $W = [w_{ij}]$ represents the weight of edges. The goal is to offer for free or advertise some of the products to a set of users $S \subseteq V$ such that through their influence on others, the revenue increases. Following Hartline et al. [2008] model, a user's value v_i^q for a product q is determined by the set of other users that own the product, i.e., $v_i^q : 2^V \to \mathbb{R}^+$. The function $v_i^q(S)$ usually takes a concave graph model [Hartline et al., 2008; Babaei et al., 2013], i.e., for all $i \in V$ and $S \subseteq V \setminus \{i\}$, we have $v_i^q(S) = g_i^q(\sum_{j\in S\cup\{i\}} w_{ij})$, where $g_i^q : \mathbb{R}^+ \to \mathbb{R}^+$ is a concave function (depending on the product $q \in Q$) and w_{ij} are chosen independently from a distribution μ. The revenue of a set S for a product $q \in Q$ is defined as

$$f^q(S) = \sum_{i\in V\setminus S} v_i^q(S) = \sum_{i\in V\setminus S} g_i^q\Big(\sum_{j\in S\cup\{i\}} w_{ij}\Big). \qquad (5)$$

Note that f^q is a non-monotone submodular function. Each product $q \in Q$ can be advertised to a potentially different subset $S^q \subseteq V$. The *total revenue*, that we try to maximize, is $\sum_{q\in Q} f^q(S^q)$ which is again a non-monotone submodular function. Now, users in a social network may want to see only a small number of advertisements. This re-

quirement can be modeled by a partition matroid. Moreover, nodes with higher (weighted) degrees are usually more influential and harder to get. So we also define a cost $c_i = c_i(\sum_{j \in V} w_{ij})$ for including a node i to a set of users targeted for advertising any product. Again, the total cost cannot exceed a threshold modeled by a knapsack.

5. Our Algorithm: FANTOM

In this section, we describe a very fast algorithm for maximizing a non-monotone submodular function subject to the intersection of a p-system and l-knapsack constraints. Our algorithm is a novel combination of two algorithms: an algorithm for maximizing *non-monotone* submodular function subject to a p-system [Gupta et al., 2010], and an algorithm for maximizing *monotone* submodular function subject to a p-system and l-knapsack constraints [Badanidiyuru and Vondrák, 2014]. Additionally we tighten the analysis of [Gupta et al., 2010] to get a better approximation ratio even for the case of $l = 0$.

Our algorithm is split into three parts. In the first part we take the most natural algorithm for maximizing submodular functions, i.e., the celebrated greedy algorithm. We restrict the greedy algorithm to only pick elements with enough "density" for knapsack constraints. In general, greedy algorithms don't tend to work well for non-monotone submodular functions. We prove that the algorithm either picks a good enough of a solution, or if we throw away the greedy solution, the optimal solution in the remaining elements is not too bad. Based on this observation, in the second part we iterate the greedy algorithm multiple times on the remaining elements to generate multiple solutions and pick the best among them. In the third and final part, we discretize and iterate over all possible values of "density" as defined in the first part.

5.1. Greedy with Density Threshold (GDT)

In the first part, we consider a natural variant of the greedy algorithm, where we pick elements in a greedy manner while simultaneously restricting it to pick elements with enough "density" for knapsack constraints. I.e., GDT (outlined in Alg. 1) does not pick elements if the ratio of the marginal value of the element to the sum of its costs for each knapsack is below a given threshold.

Theorem 5.1. *For any set $C \in \mathcal{I}$,* GDT *outputs a set $S \in \mathcal{I}$ such that*

$$f(S) \geq \min\left(\frac{\rho}{2}, \frac{1}{p+1} f(S \cup C) - \frac{l\rho}{p+1}\right).$$

5.2. Iterated Greedy with Density Threshold (IGDT)

While greedy tends to perform well for monotone functions, it can pick really bad solutions for non-monotone functions. In this part, we run GDT multiple times, each time on remaining elements to get multiple solutions. We prove that this process produces at least one reasonable solution.

Algorithm 1 GDT - Greedy with density threshold

input $f : 2^E \to \mathbb{R}_+$, a membership oracle for p-system $\mathcal{I} \subset 2^E$, and ℓ knapsack-cost functions $c_i : E \to [0, 1]$, density threshold ρ.
output A set $S \subseteq E$ satisfying $S \in \mathcal{I}$ and $c_i(S) \leq 1 \forall i$.
1: Run greedy and at each step pick the element if and only if $\frac{f_S(j)}{\sum_{i=1}^{l} c_{ij}} \geq \rho$, where $f_S(j) = f(S \cup \{j\}) - f(S)$
2: Let $z = \text{argmax}\{f(j) | j \in E\}$
3: Return $\text{argmax}(f(S), f(\{z\}))$

Algorithm 2 IGDT: Iterated greedy with density threshold

input $f : 2^E \to \mathbb{R}_+$, a membership oracle for p-system $\mathcal{I} \subset 2^E$, and ℓ knapsack-cost functions $c_i : E \to [0, 1]$, density threshold ρ.
output A set $S \subseteq E$ satisfying $S \in \mathcal{I}$ and $c_i(S) \leq 1 \forall i$.
1: $\Omega = E$
2: **for** $i = 1; i \leq p + 1; i++$ **do**
3: $S_i = \text{GDT}(f, \Omega, \rho)$
4: $S'_i = \text{Unconstrained-Maximization}(S_i)$
5: $U = U \cup \{S_i, S'_i\}$
6: $\Omega = \Omega - S_i$
7: **end for**
8: Return $\text{argmax}\{f(S) | S \in U\}$

Theorem 5.2. *For any set $C \in \mathcal{I}$,* IGDT *(outlined in Alg. 2) outputs a set $S \in \mathcal{I}$ such that*

$$f(S) \geq \min\left(\frac{\rho}{2}, \frac{p}{(p+1)(2p+1)} f(C) - \frac{l\rho}{2p+1}\right)$$

5.3. FANTOM

In this section, we consider the final piece of the puzzle. In the previous two algorithms, we consider the density threshold to be a given number. In our final algorithm we discretize the set of density thresholds into $\log(n)/\epsilon$ different possible values and run the previous algorithm on each of them. We finally show that for at least one of the discretized density thresholds we should get a good enough solution.

Theorem 5.3. FANTOM *(outlined in Alg. 3) has an approximation ratio $(1 + \epsilon)(p + 1)(2p + 2l + 1)/p$ with running time $O(\frac{nrp \log(n)}{\epsilon})$.*

Without any knapsack constraints ($l = 0$), each call to IGDT (Alg. 2) in FANTOM returns the same solution. Hence, for the case of $l = 0$, we obtain an improved ap-

Algorithm 3 FANTOM

input $f : 2^E \to \mathbb{R}_+$, a membership oracle for p-system $\mathcal{I} \subset 2^E$, and ℓ knapsack-cost functions $c_i : E \to [0, 1]$.
output A set $S \subseteq E$ satisfying $S \in \mathcal{I}$ and $c_i(S) \leq 1 \forall i$.

$M = \max_{j \in E} f(j), \gamma = \frac{2 \cdot p \cdot M}{(p+1)(2p+1)}, U = \{\}$
$R = \{\gamma, (1+\epsilon)\gamma, (1+\epsilon)^2\gamma, (1+\epsilon)^3\gamma, \ldots, \gamma \cdot n\}$
for $\rho \in R$ **do**
  $S = \text{IGDT}(f, \Omega, \rho)$
  $U = U \cup \{S\}$
end for
Return argmax$\{f(S) | S \in U\}$

proximation guarantee of $(p+1)(2p+1)/p$, with a similar running time $O(nrp)$ to [Gupta et al., 2010].

Proposition 5.4. *For the case of $l = 0$,* FANTOM *has a $(p+1)(2p+1)/p$-approximation ratio with $O(nrp)$ running time.*

As we noted in Table 1, even for the special case of 1-matroid and 1-knapsack constraints, all the existing algorithms have exorbitant running times and cannot be implemented in any reasonable time in practice. There are two main reasons for this. The first is due to an expensive enumeration step running over all subsets of very large size, and the second is due to running the continuous greedy algorithm. To compare our algorithms against practical baselines in Section 6, we consider two heuristics based on classical methods for maximizing submodular functions.

Greedy: Our first baseline starts with an empty set $S = \phi$ and keeps adding elements one by one greedily while the p-system and l-knapsack constraints are satisfied.

Density Greedy: Our second baseline starts with an empty set $S = \phi$ and keeps adding elements greedily by their value to total-knapsack cost ratio while the p-system and l-knapsack constraints are satisfied.

The above heuristics do not have provable performance guarantees as shown by the following examples.

Bad example for Greedy. Let $n = |E|$ be the number of elements in the ground set and let $m = n/2$. Define sets $T_i = \{y_i, z_i\}$ for $1 \leq i \leq m$. Let $E = \cup_{i=1}^m T_i$. Let $Y = \{y_1, y_2, \ldots, y_m\}$ and $Z = \{z_1, z_2, \ldots, z_m\}$. Let $\epsilon > 0$ be a small constant. Define the submodular function

$$\forall S \subseteq E, f(S) = (1+\epsilon) \cdot |S \cap Y| + |S \cap Z|$$

and the following two constraints.

1. A partition matroid constraint where S is a feasible solution if for $1 \leq i \leq m, |S \cap T_i| \leq 1$.
2. A knapsack constraint where cost is defined as follows. For any $e \in Y, c(e) = 1 - \frac{1}{2m}$ and for any $e \in Z, c(e) = 1/m$.

Then, it is easy to see that Baseline 1 picks a set $S = \{y_i\}$ for some i and gets value $1 + \epsilon$, while the optimal solution is Z of value $m = n/2$.

Bad example for Density Greedy. Let $T_1 = \{y_1, z_1\}$ and $T_2 = \{y_2, z_2\}$ and $E = T_1 \cup T_2$. Let $Y = \{y_1, y_2\}$ and $Z = \{z_1, z_2\}$. Let $\epsilon > 0$ be a small constant. Define the submodular function

$$\forall S \subseteq E, f(S) = \epsilon \cdot |S \cap Y| + |S \cap Z|$$

and the following two constraints.

1. A partition matroid constraint where S is a feasible solution if for $1 \leq i \leq 2, |S \cap T_i| \leq 1$,
2. A knapsack constraint where cost is defined as follows. For any $e \in Y, c(e) = \epsilon/2$ and for any $e \in Z, c(e) = 1/2$.

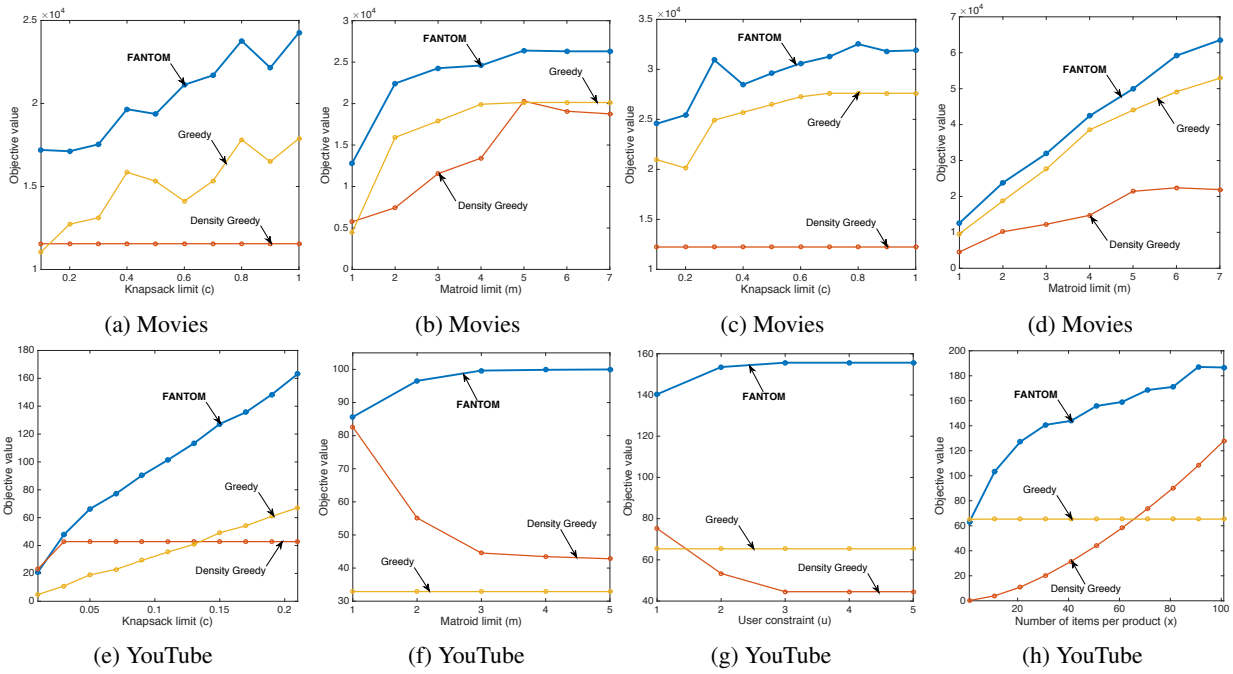
Then, it is easy to see that Baseline 2 picks a set $S = Y$ for some i and gets value 2ϵ, while the optimal solution is Z of value 2. Hence the approximation ration is at least $1/\epsilon$ and as $\epsilon \to 0$ we get unbounded approximation ratio.

6. Experiments

In this section, we evaluate FANTOM on the three real-world applications we described in Section 4: personalized movie recommendation, personalized image summarization, and revenue maximization. The main goal of this section is to validate our theoretical results, and demonstrate the effectiveness of FANTOM in practical scenarios where existing algorithms are incapable of providing desirable solutions.

Personalized movie recommendation: Our personalized recommendation experiment involves FANTOM applied to a set of 10,437 movies from the MovieLens ratings database [Mov, 2015]. Each movie is associated with a 25 dimensional feature vector calculated from user ratings. There are 19 genres in total, and each movie is associated with at most 8 genres. We used the inner product of the non-normalized feature vectors to compute the similarity $s_{i,j}$ between movies i and j (this idea was inspired by Lindgren et al. [2015]). The costs c_i are drawn from the Beta$(10, 2)$ cumulative distribution $c_i = F_{\text{Beta}(10,2)}(r_i)$, where $r_i \in (0, 1)$ is the normalized average rating of movie i. The Beta distribution lets us differentiate the highly rated movies from those with lower ratings and can be used as a proxy for the cost of watching different movies.

Fig. 1a compares the performance of our approach to the benchmarks using Eq. 2 with $\lambda = 1$ for three genres: adventure, animation, and fantasy. A total of $l = 19$ uniform



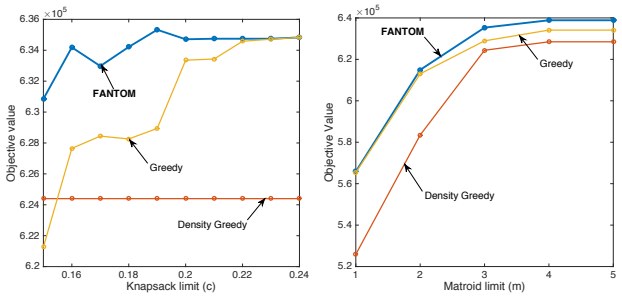
(a) Movies (b) Movies (c) Movies (d) Movies

(e) YouTube (f) YouTube (g) YouTube (h) YouTube

Figure 1. Performance of FANTOM compared to the benchmarks for movie recommendation from a set of 10,437 movies from MovieLens, and revenue maximization on top 5000 communities of YouTube with 39,841 nodes and 224,235 edges. a) shows the performance of FANTOM based on Eq. 2 for recommending movies from three genres: adventure, animation, and fantasy for $m = 3$, and varying knapsack limit c. b) shows the same quantity for $c = 1$, and varying the matroid limits m. c) shows the solution value based on Eq. 3 with $m = 3$, and varying c. d) shows the same quantity for $c = 1$, and varying m. e) shows the performance of FANTOM for selling $q = 10$ product types, with $x = 50$ available items per product, matroid limit $m = 5$ for all communities, user constraint $u = 3$, and varying knapsack limit c. f) shows the same quantity for $c = 0.1$, $q = 10$, $x = 50$, $u = 3$ and varying m. g) shows the solution value for $c = 0.2$, $q = 10$, $x = 50$, $m = 3$, and varying u. h) shows the same quantity for $c = 0.2$, $q = 10$, $m = 5$, $u = 3$ and varying x.

matroid constraints are considered to limit the number of movies chosen from each of the 19 genres. The limits for all the matroid constraints are set to 3. We also considered an additional uniform matroid constraint to restrict the size of the final solution to 10 movies. Moreover, a knapsack constraint is considered to model the total available budget. It can be seen that FANTOM significantly outperforms the benchmarks for different knapsack limits. Fig. 1b shows similar qualitative behavior for a fixed knapsack limit $c = 1$, and varying matroid limits m associated with each of the 19 genres. Again, FANTOM is able to show a good performance in scenarios where Greedy and Density Greedy perform arbitrary poorly. Fig. 1c and 1d show the same qualitative behavior using Eq. 3. Table 2 summarized the movies recommended by different methods, along with their average rating and associated genres. We can see that by using Eq. 3 the recommended movies have more common genres with what the user requested.

Revenue maximization with multiple products: Our larger scale experiment involves applying FANTOM to maximize the revenue function defined in Eq. 5. We performed our experiment on the top 5000 largest communities of the YouTube social network consists of 39,841 nodes and 224,235 edges [Yang and Leskovec, 2015]. We consider the settings where we are to advertise $|Q| = q$ different types of product across all communities of the same social network. For simplicity, we assume that there are x units available from each product, and the influence of individuals on each other is the same for all product types. The



(a) Images (b) Images

Figure 3. Performance of FANTOM compared to the benchmarks for personalized image summarization: a) shows the solution value for summarizing three categories airplane, automobile, and bird for $m = 3$, and varying the knapsack limit c. b) shows the same quantity for $c = 0.1$, and varying the matroid limits m.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Eq.	FANTOM			Greedy			Density Greedy		
	Movie	Average rate	Genres	Movie	Average rate	Genres	Movie	Average rate	Genres
2	1	3.99	**1,2**,15	6	4.20	**2**,3,**9**	11	1.16	3,4
	2	3.98	3,4,5,**9**	7	4.21	**1,2**	12	1.16	**2**,4
	3	4.00	**2**,3,4,**9**,19	8	3.26	**1,2**,8,14	13	1.25	**2**,4,5,**9**
	4	3.08	**9**,11	9	2.66	**2**,5,12	14	1.34	**2**,3,4,**9**
	5	2.53	3,4,5	10	1.96	**1,2**,4	15	1.39	**1,2**,4
3	16	3.91	**2**,3,5,**9**,14	21	3.78	**1,2**,3,4,5,**9**	12	1.16	**2**,4
	17	3.70	**2**,3,4,8,**9**	22	3.75	**1,2**,3,4,5,**9**	13	1.25	**2**,4,5,**9**
	18	3.78	**1,2**,5,12,14,16	23	4.06	**1,2**,3,4,**9**,15	14	1.34	**2**,3,4,**9**
	19	3.53	**1,2**,3,4,5	24	3.82	**2**,5,8,**9**,13,15,16	11	1.16	3,4
	20	3.16	**1,2**,5,**9**,11,16	25	2.08	**1,2**,4,5,**9**,15	15	1.39	**1,2**,4

Table 2. Movies recommended by FANTOM vs. Greedy and Density Greedy using Eq. 2, and Eq. 3 for $m = 5$ and $c = 1$. There are 19 genres in total: Action(1), Adventure(2), Animation (3), Children (4), Comedy (5), Crime (6), Documentary (7), Drama (8), Fantasy (9), Film-Noir (10), Horror (11), Musical (12), Mystery (13), Romance (14), Sci-Fi (15), Thriller (16), War (17), Western (18), IMAX (19). The user is interested in adventure, animation, and fantasy movies (genres **1,2,9**). See the Appendix for a complete list of movie names.

edge weights are assigned according to a uniform distribution $\mathcal{U}(0, 1)$, and the cost of selecting each node c_i is determined according to an exponential cumulative distribution function of the normalized sum of its edge weights. For the exponential distribution, we chose the parameter $\lambda = 0.2$ to scale the costs to the interval $[0, 1]$. To model different characteristics of the products, we model the revenues by the concave function $v_i^q(S) = \alpha_q \sqrt{\sum_{j \in S} w_{i,j}}$, where α_q depends on the type of the product. We used $q = 10$ different values $\alpha_q \in [0.8, 1.3]$ to model the revenue of different product types. Finally, we modeled user constraints by a partition matroid that puts a limits u on the number of products that can be offered to each user. Another partition matroid is employed to restrict the number of products m offered for free to users in each community.

Fig. 1e shows the revenue obtained by FANTOM versus the budget c when there are $x = 50$ free items available from each product, the number of individuals that can be selected from each community is limited to $m = 5$, and the number of products that can be offered to each user is at most $u = 3$. We note again that FANTOM significantly outperforms the other benchmarks. Fig. 1f shows the same behavior for varying the matroid limit m, when $x = 50$ and budget $c = 0.1$. Similarly, Fig. 1g shows the performance of FANTOM for $m = 5$, $x = 50$, $c = 0.2$, and varying the user constraints u. Finally, Fig. 1h shows the performance of FANTOM for $m = 5$, $u = 3$, $c = 0.2$, and varying the number of available items x from each product type.

Personalized image summarization: Our personalized recommendation experiment involves FANTOM applied to Eq. 4. We performed our experiments on a set of 10,000 Tiny Images [Krizhevsky and Hinton, 2009]. The images belong to 10 classes, with 1000 images per class. Each 32 by 32 RGB pixel image was represented by a 3,072 dimensional vector. We used the inner product to compute the similarity $s_{i,j}$ between image i and j. The costs are chosen proportional to the normalized variance of the image pixels as a simple technique to calculate image qualities. This way, we assign a higher cost to images with higher contrast and a lower cost to blurry images.

A partition matroid constraint is considered to limit the number of images chosen from each of the specified categories. Moreover, a knapsack constraint is employed to model the limited available budget. Fig. 3a compares the performance of our approach to the benchmarks for summarizing images from three categories: airplane, automobile, and bird. The results are shown for varying knapsack limit c, while the maximum number of images allowed from each category is set to $m = 3$. Similarly, Fig. 3b shows the results for fixed $c = 0.1$, and varying m. We find again that FANTOM significantly outperforms the benchmarks.

7. Conclusion

We have developed the first efficient algorithm FANTOM for maximizing non-monotone submodular functions subject to a p-system and l-knapsack constraints. We have also showed its applications to various personalized data summarization problems. Given the importance of submodular optimization to numerous data mining and machine learning applications, we believe our results provide an important step towards addressing various constrained discrete optimization problems.

Acknowledgments. This research was supported by a Google Faculty Research Award.

References

Grouplens. movielens 20m dataset. http://grouplens.org/datasets/movielens/20m/, 2015.

Mahmoudreza Babaei, Baharan Mirzasoleiman, Mahdi Jalili, and Mohammad Ali Safari. Revenue maximization in social networks through discounting. *Social Network Analysis and Mining*, 3(4):1249–1262, 2013.

Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *SODA*, 2014.

Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: massive data summarization on the fly. In *KDD*, 2014.

Niv Buchbinder, Moran Feldman, Joseph Seffi, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM Journal on Computing*, 2015.

Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 2009.

Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: Matchings, matroids, and more. *Mathematical Programming*, 2015.

Chandra Chekuri and Martin Pál. A recursive greedy algorithm for walks in directed graphs. In *FOCS*, 2005.

Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM J. Comput.*, 2014.

Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *Automata, Languages, and Programming*. 2015a.

Chandra Chekuri, TS Jayram, and Jan Vondrák. On multiplicative weight updates for concave and submodular function maximization. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 201–210. ACM, 2015b.

Anirban Dasgupta, Ravi Kumar, and Sujith Ravi. Summarization through submodularity and dispersion. In *ACL*, 2013.

Delbert Dueck and Brendan J Frey. Non-metric affinity propagation for unsupervised image categorization. In *ICCV*, 2007.

Khalid El-Arini and Carlos Guestrin. Beyond keyword search: discovering relevant scientific literature. In *KDD*, 2011.

Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In *GKDD 2009*, 2009.

Uriel Feige, Vahab S Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 2011.

Moran Feldman, Joseph Naor, and Roy Schwartz. Non-monotone submodular maximization via a structural continuous greedy algorithm - (extended abstract). In *ICALP (1)*, pages 342–353, 2011.

Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Near-optimal MAP inference for determinantal point processes. In *NIPS*, 2012.

Ryan Gomes and Andreas Krause. Budgeted nonparametric learning from data streams. In *ICML*, 2010.

Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *WINE*, 2010.

Anupam Gupta, Viswanath Nagarajan, and R Ravi. Robust and maxmin optimization under matroid and knapsack uncertainty sets. *ACM Transactions on Algorithms (TALG)*, 12(1):10, 2015.

Jason Hartline, Vahab Mirrokni, and Mukund Sundararajan. Optimal marketing strategies over social networks. In *WWW*. ACM, 2008.

David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

Andreas Krause and Carlos Guestrin. Submodularity and its applications in optimized information gathering. *ACM TIST*, 2011.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. Fast greedy algorithms in mapreduce and streaming. *ACM Transactions on Parallel Computing*, 2015.

Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *STOC*, 2009.

Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Math. Oper. Res.*, 2010.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne M. VanBriesen, and Natalie S. Glance. Cost-effective outbreak detection in networks. In *KDD*, 2007.

Hui Lin and Jeff A. Bilmes. A class of submodular functions for document summarization. In *ACL*, 2011.

Erik M Lindgren, Shanshan Wu, and Alexandros G Dimakis. Sparse and greedy: Sparsifying submodular facility location problems. 2015.

Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *NIPS*, 2013.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *AAAI*, 2015.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions - i. *Math. Prog.*, 14:265–294, 1978.

Colorado Reed and Zoubin Ghahramani. Scaling the indian buffet process via submodular maximization. In *ICML*, 2013.

Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

Ian Simon, Noah Snavely, and Steven M Seitz. Scene summarization for online image collections. In *ICCV*, 2007.

Adish Singla, Ilija Bogunovic, Gábor Bartók, Amin Karbasi, and Andreas Krause. Near-optimally teaching the crowd to classify. *ICML*, 2014.

Ruben Sipos, Adith Swaminathan, Pannaga Shivaswamy, and Thorsten Joachims. Temporal corpus summarization using submodular word coverage. In *CIKM*, 2012.

Sebastian Tschiatschek, Rishabh Iyer, Haochen Wei, and Jeff Bilmes. Learning mixtures of submodular functions for image collection summarization. In *NIPS*, 2014.

Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, 2008.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. Fast multi-stage submodular maximization. In *ICML*, 2014.

Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.