ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200123

Non-Monotone Submodular Maximization with Multiple Knapsacks in Static and Dynamic Settings[1]

Vanja Doskoč[2] and **Tobias Friedrich**[3] and **Andreas Göbel**[4] and **Aneta Neumann**[5] and **Frank Neumann**[6] and **Francesco Quinzan**[7]

Abstract. We study the problem of maximizing a non-monotone submodular function under multiple knapsack constraints. We propose a simple discrete greedy algorithm to approach this problem, and prove that it yields strong approximation guarantees for functions with bounded curvature. In contrast to other heuristics, this does not require problem relaxation to continuous domains and it maintains a constant-factor approximation guarantee in the problem size. In the case of a single knapsack, our analysis suggests that the standard greedy can be used in non-monotone settings.

Additionally, we study this problem in a dynamic setting, in which knapsacks change during the optimization process. We modify our greedy algorithm to avoid a complete restart at each constraint update. This modification retains the approximation guarantees of the static case.

We evaluate our results experimentally on a video summarization and sensor placement task. We show that our proposed algorithm competes with the state-of-the-art in static settings. Furthermore, we show that in dynamic settings with tight computational time budget, our modified greedy yields significant improvements over starting the greedy from scratch, in terms of the solution quality achieved.

1 INTRODUCTION

Many artificial intelligence and machine learning tasks can be naturally approached by maximizing submodular objectives. Examples include subset selection [10], document summarization [25], video summarization [27] and action recognition [38]. Submodular functions are set functions that yield a diminishing return property: it is more helpful to add an element to a smaller collection than to add it to a larger one. This property fully characterizes the notion of submodularity.

Practical applications often require additional side constraints on the solution space, determined by possible feasibility conditions. These constraints can be complex [25, 29, 36]. For instance, when performing video summarization tasks, we might want to select frames that fulfill costs constraints based on qualitative factors, such as resolution and luminance.

In this paper, we study general multiple knapsack constraints. Given a set of solutions, a k-knapsack constraint consists of k linear cost functions c_i on the solution space and corresponding weights W_i. A solution is then feasible if the corresponding costs do not exceed the weights. In this paper, we study the problem of maximizing a submodular function under a k-knapsack constraint.

Sometimes, real-world optimization problems involve dynamic and stochastic constraints [6]. For instance, resources and costs can exhibit slight frequent changes, leading to changes of the underlying space of feasible solutions. Various optimization problems have been studied under dynamically changing constraints, i.e., facility location problems [18], target tracking [14], and other submodular maximization problems for machine learning [4]. Motivated by these applications, we also study the problem of maximizing a submodular function under a k-knapsack constraint, when the set of feasible solutions changes online.

Literature Overview. Khuller, Moss and Naor [19] show that a simple greedy algorithm achieves a $1/2(1-1/e)$-approximation guarantee, when maximizing a modular function with a single knapsack constraint. They also propose a modified greedy algorithm that achieves a $(1-1/e)$-approximation. Sviridenko [33] shows that this modified greedy algorithm yields a $(1-1/e)$-approximation guarantee for monotone submodular functions under a single knapsack constraint. Its run time is $O(n^5)$ function evaluations. The optimization of monotone submodular functions under a given chance constraint by an adaptation of these simple greedy algorithm has recently been investigated by Doerr et al. [11].

Lee et al. [24] give a $(1/5-\varepsilon)$-approximation local search algorithm, for maximizing a non-monotone submodular function under multiple knapsack constraints. Its run time is polynomial in the problem size and exponential in the number of constraints. Fadaei, Fazli and Safari [13] propose an algorithm that achieves a $(1/4-\varepsilon)$-approximation algorithm for non-monotone functions. This algorithm requires to compute fractional solutions of a continuous extension of the value oracle function f. Chekuri, Vondrák and Zenklusen [5] improve the approximation ratio to $0.325-\varepsilon$, in the case of $k = \mathcal{O}(1)$ knapsacks. Kulik, Schachnai and Tamir [23] give a $(1-1/e-\varepsilon)$-approximation algorithm when f is monotone and a $(1/e-\varepsilon)$-approximation algorithm when the function is non-monotone. Again, their method uses continuous relaxations of the discrete setting. FANTOM is a popular algorithm for non-monotone submodular maximization [28]. It can handle intersections of a variety of constraints. In the case of multiple knapsack constraints, it achieves a $1/((1+\varepsilon)(10+4k))$-approximation in $\mathcal{O}\left(n^2\log(n)/\varepsilon\right)$ run time.

Submodular optimization problems with dynamic cost constraints, including knapsack constraints, are investigated in [31, 32]. Rostapoor et al. [32] show that a Pareto optimization approach can implicitly

[1] A version of this paper with appendix can be found in [12].
[2] Hasso Plattner Institute, email: vanja.doskoc@hpi.de
[3] Hasso Plattner Institute, email: friedrich@hpi.de
[4] Hasso Plattner Institute, email: andreas.goebel@hpi.de
[5] The University of Adelaide, email: aneta.neumann@adelaide.edu.au
[6] The University of Adelaide, email: frank.neumann@adelaide.edu.au
[7] Hasso Plattner Institute, email: francesco.quinzan@hpi.de

deal with dynamically changing constraint bounds, whereas a simple adaptive greedy algorithm fails.

Our Contribution. Many of the aforementioned algorithmic results, despite having polynomial run time, seem impractical for large input applications. Following the analysis outlined in [8, 16, 19], we propose a simple and practical discrete algorithm to maximize a submodular function under multiple knapsack constraints. This algorithm, which we call the λ-GREEDY, achieves a $(1-e^{-1/\lambda})/(3\max(1,\alpha))$-approximation guarantee on this problem, with α expressing the curvature of f, and $\lambda \in [1,k]$ a constant. It requires at most $\mathcal{O}(n^{\max(k/\lambda,2)})$ function evaluations. To our knowledge this is the first algorithm yielding a trade-off between run-time and approximation ratio.

We also propose a robust variation of our λ-GREEDY, which we call λ-DGREEDY, to handle dynamic changes in the feasibility region of the solution space. We show that this algorithm maintains a $(1-e^{-1/\lambda})/(3\max(1,\alpha))$-approximation without having to do a complete restart of the greedy sequence.

We demonstrate experimentally that our algorithms yield good performance in practise, with two real-world scenarios. First, we consider a video summarization task, which consists of selecting representative frames of a given video [29, 28]. We also consider a sensor placement problem, that asks to select informative thermal stations over a large territory [20].

Our experiments indicate that the λ-GREEDY yields superior performance to commonly used algorithms for the static video summarization problem. We then perform experiments in dynamic settings with both scenarios, to show that the robust variation yields significant improvement in practise.

The paper is structured as follows. In Section 2 we introduce basic definitions and define the problem. In Section 3 we define the algorithms. We present the theoretical analysis in Section 4, and the experimental framework in Section 5. The experimental results are discussed in Section 6 and Section 7. We conclude in Section 8.

2 PRELIMINARIES

Submodularity and Curvature In this paper, we consider problems with an *oracle function* that outputs the quality of given solution. We measure performance in terms of calls to this function, since in many practical applications they are difficult to evaluate. We assume that value oracle functions are submodular, as in the following definition.

Definition 1 (Submodularity) *Given a finite set V, a set function $f\colon 2^V \to \mathbb{R}$ is submodular if for all $S,\Omega \subseteq V$ we have that $f(S)+f(\Omega) \geq f(S\cup\Omega)+f(S\cap\Omega)$.*

For the equivalent intuitive definition described informally in the introduction see [30].

For any submodular function $f\colon 2^V \to \mathbb{R}$ and sets $S,\Omega \subseteq V$, we define the *marginal value* of Ω with respect to S as $f_S(\Omega) = f(S\cup\Omega)-f(S)$. Note that, if f only attains non-negative values, it holds that $f(\Omega) \geq f_S(\Omega)$ for all $S,\Omega \subseteq V$.

Our approximation guarantees use the notion of *curvature*, a parameter that bounds the maximum rate with which a submodular function changes. We say that a submodular function $f\colon 2^V \to \mathbb{R}_{\geq 0}$ has curvature α if the value $f(S\cup e)-f(S)$ does not change by a factor larger than $1-\alpha$ when varying S, for all $e \in V \setminus S$. This parameter was first introduced by [8] and later revisited in [3]. We use the following definition of curvature, which slightly generalizes that proposed in Friedrich et al. [16].

Definition 2 (Curvature) *Let $f\colon 2^V \to \mathbb{R}_{\geq 0}$ be a submodular set function. The curvature is the smallest scalar α such that*

$$f_\omega((S\cup\Omega)\setminus\{\omega\}) \geq (1-\alpha) f_\omega(S\setminus\{\omega\}),$$

for all $S,\Omega \subseteq V$ and $\omega \in S\setminus\Omega$.

Note that $\alpha \leq 1$ always holds and that all monotone submodular functions have curvature always bounded as $0 \leq \alpha \leq 1$. It follows that all submodular functions with negative curvature are non-monotone.

Problem Description The problem of maximizing a submodular function under multiple knapsack constraints can be formalized as follows.

Problem 3 *Let $f\colon 2^V \to \mathbb{R}_{\geq 0}$ be a submodular function.*[8] *Consider linear cost functions $c_i\colon 2^V \to \mathbb{R}_{\geq 0}$,*[9] *and corresponding weights W_i, for all $i \in [k]$. We search for a set* OPT $\subseteq V$, *such that* OPT $\in \arg\max_{S\subseteq V}\{f(S)\colon c_i(S) \leq W_i, \forall i \in [k]\}$.

In this setting, one has k knapsacks and wishes to find an optimal set of items such that its total cost, expressed by the functions c_i, does not violate the capacity of each knapsack. Note that the same set might have different costs for different knapsacks.

We denote with (c,W) the constraint requirements $c_i(S) \leq W_i$ for all $S \subseteq V$, for all $i \in [k]$. For a ground set V with a fixed ordering on the points and a cost function c_i, where $i \in [k]$, let τ_i be a permutation on V where $c_i(e_{\tau_i(1)}) \geq \cdots \geq c_i(e_{\tau_i(n)})$. We define the value $\chi(c,W)$ as

$$\chi(c,W) = \min_{i\in[k]} \arg\max_{j\in[|V|]} \left\{ \sum_{\ell=1}^{j} c_i(e_{\tau_i(\ell)}) \leq W_i \right\}.$$

Note that the value $\chi(c,W)$ is such that each set $U \subseteq V$ with cardinality $|U| \leq \chi(c,W)$ is feasible for all constraints in (c,W).[10]

We observe that in the case of a single knapsack, if $c_1(S) = |S|$ for all $S \subseteq V$, then Problem 3 consists of maximizing a submodular function under a cardinality constraint, which is known to be NP-hard.

In our analysis we always assume that the following reduction holds.

Reduction 4 *For Problem 3 we may assume that there exists a point $e^* \in V$ such that $f(S\cup e^*) = f(S)$ for all $S \subseteq V$, and $c_i(e^*) = W_i$ for all $i \in [k]$. Furthermore, we may assume that $c_i(e) \leq W_i$ for all $e \in V$, for all $i \in [m]$.*

If the conditions of Reduction 4 do not hold, one can remove all points $e \in V$ that violate one of the constraints and add a point e^* without altering the function f. Intuitively, Reduction 4 requires that each singleton, except for one, is feasible for all knapsack constraints. This ensures that $\arg\max_{e\in V} f(e)$ is always feasible in all constraints, since $f(e^*) = 0$, and the optimum solution consists of at least one point. Furthermore, the point $e^* \in V$ ensures that the solution quality never decreases throughout a greedy optimization process, until a non-feasible solution is reached.

[8] We assume that $f(\emptyset) = 0$, and that f is non-constant.

[9] We assume that $\max_j c_j(e) > 0$ for all $e \in V$.

[10] We remark that for our purposes, the value $\chi(c,W)$ could be defined directly as the value where each set $U \subseteq V$ with cardinality $|U| \leq \chi(c,W)$ is feasible under the constraints in (c,W). However, this value is in general NP-hard to compute [7, 35].

Algorithm 1: The λ-GREEDY algorithm.

input: submodular function f, k-knapsacks (c, W) and parameter λ ;
output: an approximate feasible global maximum of f;
$\mathcal{V} \leftarrow \{e \in V : c_j(e) \leq \lambda W_j/k \; \forall j \in [k]\}$;
$S \leftarrow \mathcal{V}$;
$\sigma \leftarrow \emptyset$;
$v^* \leftarrow \arg\max_{e \in V} f(e)$;
while $S \neq \emptyset$ **do**
  let $e \in S$ maximizing $f_\sigma(e)/\max_j c_j(e)$;
  $S \leftarrow S \setminus e$;
  if $c_j(\sigma \cup e) \leq W_j, \forall j \in [k]$ **then** $\sigma \leftarrow \sigma \cup e$;
return $\arg\max_{\{U \subseteq V \setminus \mathcal{V} : \, c_i(U) \leq W_i \; \forall i\}} \{f(\sigma), f(v^*), f(U)\}$;

Algorithm 2: The λ-DGREEDY algorithm.

input: submodular function f, k-knapsacks (c, W) with dynamic weights and parameter λ;
output: an approximate feasible global maximum of f;
evaluate f over all sets of cardinality at most k/λ;
$\mathcal{V} \leftarrow \{e \in V : c_j(e) \leq \lambda W_j/k \; \forall j \in [k]\}$;
$S \leftarrow \mathcal{V}$;
$\sigma \leftarrow \emptyset$;
$v^* \leftarrow \arg\max_{e \in V} f(e)$;
while $S \neq \emptyset$ **do**
  // greedy rule
  let $e \in S$ maximizing $f_\sigma(e)/\max_j c_j(e)$;
  $S \leftarrow S \setminus e$;
  if $c_i(\sigma \cup e) \leq W_i, \forall i \in [k]$ **then** $\sigma \leftarrow \sigma \cup v$;
  // update rule
  if *new weights* $\{W_i'\}$ *are given* **then**
    $\mathcal{V}' \leftarrow \{e \in V : c_j(e) \leq \lambda W_j'/k \; \forall j \in [k]\}$;
    while $|\sigma| > \min\{\chi(c, W), \chi(c, W')\} \vee \sigma \not\subseteq \mathcal{V} \cap \mathcal{V}'$ **do**
      let $e \in \sigma$ be the last point added to σ;
      $\sigma \leftarrow \sigma \setminus e$;
    $\mathcal{V} \leftarrow \mathcal{V}'$;
    $S \leftarrow \mathcal{V} \setminus \sigma$;
return $\arg\max_{\{U \subseteq V \setminus \mathcal{V} : \, c_i(U) \leq W \; \forall i\}} \{f(\sigma), f(v^*), f(U)\}$;

Additionally, we study a dynamic setting of Problem 3, in which weights W_i are repeatedly updated throughout the optimization process, while the corresponding cost functions c_i remain unchanged. In this setting, we assume that an algorithm queries a function to retrieve the weights W_i which are, sometimes, updated online. We assume that weights changes occur independently of the optimization process and algorithmic operations. Furthermore, we assume that Reduction 4 holds for each dynamic update.

3 ALGORITHMS

We approach Problem 3 with a discrete algorithm based on a greedy technique, commonly used to maximize a submodular function under a single knapsack constraint (see [19, 37]). Given the parameter value $\lambda \in [k]$, our algorithm defines the following partition of the objective space:

- the set $\mathcal{V}$ containing all $e \in V$ such that $c_j \leq \lambda W_j/k$ for all $j \in [k]$;
- the complement $V \setminus \mathcal{V}$ containing all $e \in \mathcal{V}$ such that $c_j(e) > \lambda W_j/k$ for all $j \in [k]$.

The λ-GREEDY optimizes f over the set $\mathcal{V}$, with a greedy update that depends on all cost functions c_j. After finding a greedy approximate solution σ, the λ-GREEDY finds the optimum τ among feasible subsets of $V \setminus \mathcal{V}$. This step can be performed with a deterministic search over all possible solutions, since the space $V \setminus \mathcal{V}$ always has bounded size. The λ-GREEDY outputs the set with highest f-value among σ, τ or the maximum among the singletons.

From the statement of Theorem 5 we observe that the parameter λ sets a trade-off between solution quality and run time. For small λ, Algorithm 1 yields better approximation guarantee and worse run time, than for large λ. This is due to the fact that the size of $\mathcal{V} \setminus V$ depends on this parameter. In practise, the parameter λ allows to find the right trade-off between solution quality and run time, depending on available resources. Note that in the case of a single knapsack constraint, for $\lambda = k$ the λ-GREEDY is equivalent to the greedy algorithm studied in [19].

We modify the λ-GREEDY to handle dynamic constraints where weights change overtime. This algorithm, which we refer to as the λ-DGREEDY, is presented in Algorithm 2. It consists of two subroutines, which we call the *greedy rule* and the *update rule*. The greedy rule of the λ-DGREEDY uses the same greedy update as the λ-GREEDY does: At each step, find a point $v \in V$ that maximizes the marginal gain over maximum cost, and add v to the current solution, if the resulting set is feasible in all knapsacks. The update rule allows to handle possible changes to the weights, even when the greedy procedure has not terminated, without having to completely restart the algorithm.

Following the notation of Algorithm 2, if new weights $W_1', \ldots, W_k'$ are given, then the λ-DGREEDY iteratively removes points from the current solution, until the resulting set yields $\sigma \leq \min\{\chi(c, W), \chi(c, W')\}$ and $\sigma \subseteq \mathcal{V} \cap \mathcal{V}'$. This is motivated by the following facts:

1. every set $U \leq \min\{\chi(c, W), \chi(c, W')\}$ is feasible in both the old and the new constraints;
2. every set $U \leq \min\{\chi(c, W), \chi(c, W')\}$ such that $U \subseteq \mathcal{V} \cap \mathcal{V}'$ obtained with a greedy procedure yields the same approximation guarantee in both constraints;
3. every set $U \subseteq \mathcal{V} \cap \mathcal{V}'$ is such that $c_i(e) \leq \lambda W_i'/k$ for all $i \in k$, for all $e \in U$.

All three conditions are necessary to ensure that the approximation guarantee is maintained.

Note that the update rule in Algorithm 2 does not backtrack the execution of the algorithm until the resulting solution is feasible in the new constraint, and then adds elements to the current solution. For instance, consider the following example, due to Roostapour et al. [32]. We are given a set of $n+1$ items $\{e_1, \ldots, e_{n+1}\}$ under a single knapsack (c, W), with the cost function c defined as

$$c(e_i) = \begin{cases} 1, & \text{if } 1 \leq i \leq n/2 \text{ or } i = n+1; \\ 2, & \text{otherwise;} \end{cases}$$

and f-values defined on the singleton as

$$f(e_i) = \begin{cases} 1/n, & \text{if } 1 \leq i \leq n/2; \\ 1, & \text{if } n/2 < i \leq n; \\ 3, & \text{if } i = n+1; \end{cases}$$

We define $f(U) = \sum_{e \in U} f(e)$, for all $U \subseteq \{e_1, \ldots, e_{n+1}\}$. Consider Algorithm 2 optimizing f from scratch with $W = 2$. We only

consider the case $\lambda = 1$, since we only consider a single knapsack constraint. Then both Algorithm 2 and the λ-GREEDY return a set of the form $\{e_{n+1}, e_j\}$ with $1 \leq j \leq n/2$. Suppose now that the weight dynamically changes to $W = 3$. Then backtracking the execution and adding points to the current solution would result into a solution of the form $\{e_{n+1}, e_i, e_j\}$ with $1 \leq i, j \leq n/2$ with $f(\{e_{n+1}, e_i, e_j\}) = 3 + 2/n$. However, in this case it holds $\min\{\chi(c, 2), \chi(c, 3)\} = 1$, since there exists a solution of cardinality 2 that is not feasible in $(c, 2)$. Hence, Algorithm 2 removes the point e_j from the solution $\{e_{n+1}, e_j\}$, before adding new elements to it. Afterwards, it adds a point $e_{j'}$ with $n/2 < j' \leq n$ to it, obtaining a solution such that $f(\{e_{n+1}, e_{j'}\}) = 3 + 1 = 4$.
We remark that on this instance, the λ-GREEDY with a simple backtracking operator yields arbitrarily bad approximation guarantee, as discussed in [32, Theroem 3]. In contrast, Algorithm 2 maintains the approximation guarantee on this instance (see Theorem 6).

4 APPROXIMATION GUARANTEES

We prove that Algorithm 1 yields a strong approximation guarantee, when maximizing a submodular function under k knapsack constraints in the static case. This part of the analysis does not consider dynamic weight updates. We use the notion of curvature as in Definition 2.

Theorem 5 *Let f be a submodular function with curvature α, suppose that k knapsacks are given. For all $\lambda \in [1, k]$, the λ-GREEDY is a $(1 - e^{-1/\lambda})/(3\max(1, \alpha))$-approximation algorithm for Problem 3. Its run time is $\mathcal{O}\left(n^{\max(k/\lambda, 2)}\right)$.*

A proof of this result is given in the full version [12], and it is based on the work of Khuller et al. [19]. Note that if the function f is monotone, then the approximation guarantee given in Theorem 5 matches well-known results [19]. We remark that non-monotone functions with bounded curvature are not uncommon in practise. For instance, all cut functions of directed graphs are non-monotone, submodular and have curvature $\alpha \leq 2$, as discussed in [16].

We perform the run time analysis for the λ-GREEDY in dynamic settings, in which weights $\{W_i\}_i$ change over time.

Theorem 6 *Consider Algorithm 2 optimizing a submodular function with curvature $\alpha > 0$ and knapsack constraints (c, W). Suppose that at some point during the optimization process new weights W_i' are given. Let σ be the current solution before the weights update, and let $\sigma_t \subseteq \sigma$ consist of the first t points added to σ. Furthermore, let t^* be the largest index such that $\sigma_{t^*} \subseteq \mathcal{V} \cap \mathcal{V}'$, with $\mathcal{V}, \mathcal{V}'$ as in Algorithm 2, and define $\chi = \min\{\chi(c, W), \chi(c, W'), t^*\}$. Then after additional $\mathcal{O}(n(n - \chi))$ run time the λ-GREEDY finds a $(1 - e^{-1/\lambda})/(3\max(1, \alpha))$-approximate optimal solution in the new constraints, for a fixed parameter $\lambda \in [1, k]$.*

A proof of this result is given in the full version [12]. Intuitively, the proof shows that, given two sets of feasible solutions $\mathcal{V}, \mathcal{V}'$, the λ-GREEDY follows the same paths on both problem instances, for all solutions with size up to χ. Note that Theorem 5 yields the same theoretical approximation guarantee as Theorem 6. Hence, if dynamic updates occur at a slow pace, it is possible to obtain identical results by restarting λ-GREEDY every time a constraint update occurs. However, as we show in Section 7, there is significant advantage in using the λ-DGREEDY in settings when frequent noisy constraints updates occur.

5 APPLICATIONS

In this section we present a high-level overview of two possible applications for Problem 3. We describe experimental frameworks and implementations for these applications in Sections 6-7.

Video Summarization. Determinantal Point Process (DPP), [26], is a probabilistic model, the probability distribution function of which can be characterized as the determinant of a matrix. More formally, consider a sample space $V = [n]$, and let L be a positive semidefinite matrix. We say that L defines a DPP on V, if the probability of an event $S \subseteq V$ is given by

$$\mathcal{P}_L(S) = \frac{\det(L_S)}{\det(L + I)},$$

where $L_S = (L_{i,j})_{i,j \in S}$ is the submatrix of L indexed by the elements in S, and I is the $n \times n$ identity matrix. For a survey on DPPs and their applications see [22].

We model this framework with a matrix L that describes similarities between pairs of frames. Intuitively, if L describes the similarity between two frames, then the DPP prefers diversity.
In this setting, we search for a set of features $S \subseteq V$ such that $\mathcal{P}(S)$ is maximal, among sets of feasible solutions defined in terms of a knapsack constraint. Since L is positive semidefinite, the function $\log\det L_S$ is submodular [22].

Sensor Placement. The maximum entropy sampling problem consists of choosing the most informative subset of random variables subject to side constraints. In this work, we study the problem of finding the most informative set among given Gaussian time series.

Given a sample covariance matrix Σ of the time series corresponding to measurements, the entropy of a subset of sensors is then given by the formula

$$f(S) = \frac{1 + \ln(2\pi)}{2} |S| + \frac{1}{2} \ln\det(\Sigma_S)$$

for any indexing set $S \subseteq \{0, 1\}^n$ on the variation series, where $\det(\Sigma_S)$ returns the determinant of the sub-matrix of Σ indexed by S. It is well-known that the function f is non-monotone and submodular. Its curvature is bounded as $\alpha \leq 1 - 1/\mu$, where μ is the largest eigenvalue of Σ [34, 20, 16].
We consider the problem of maximizing the entropy f under a *partition matroid constraint*. This additional side constraint requires upper-bounds on the number of sensors that can be chosen in given geographical areas.

6 STATIC EXPERIMENTS

The aim of these experiments is to show that the λ-GREEDY yields better performance in comparison with FANTOM [27], which is a popular algorithm for non-monotone submodular objectives under complex sets of constraints. We consider video summarization tasks as in Section 5.

Let L be the matrix describing similarities between pairs of frames, as in Section 5. Following [17], we parametrize L as follows. Given a set of frames, let $\boldsymbol{f}_i$ be the feature vector of the i-th frame. This vector encodes the contextual information about frame i and its representativeness of other items. Then the matrix L can be paramterized as $L_{i,j} = \boldsymbol{z}_i^T W^T W \boldsymbol{z}_j$, with $\boldsymbol{z}_i = \tanh(U\boldsymbol{f}_i)$ being a hidden representation of $\boldsymbol{f}_i$, and U, W parameters. We use a neural network to

Table 1. We consider 20 movies from the Frames Labeled In Cinema dataset [15]. For each movie, we select a representative set of frames, by maximizing a submodular function under additional knapsack constraints, as described in Section 6. We run the λ-GREEDY and the FANTOM algorithm [27] with various parameter choices, until no remaining point in the search space yields improvement on the fitness value, without violating side constraints. We observe that in all cases the λ-GREEDY yields better run time and solution quality than the FANTOM.

run time for each video clip

algorithm	parameter	video (1)	video (2)	video (3)	video (4)	video (5)	video (6)	video (7)
FANTOM	$\varepsilon = 0.001$	4085317	3529230	3813637	2986719	3368901	3442329	3082814
FANTOM	$\varepsilon = 0.01$	406960	351321	382342	299444	336670	344619	307726
FANTOM	$\varepsilon = 0.1$	41008	34309	37232	29249	33343	33146	30235
λ-GREEDY	$\lambda = k$	2895	2895	2895	2709	2895	2709	2522

algorithm	parameter	video (8)	video (9)	video (10)	video (11)	video (12)	video (13)	video (14)
FANTOM	$\varepsilon = 0.001$	3171467	3781141	3121379	3673776	3603787	3055122	4119203
FANTOM	$\varepsilon = 0.01$	320738	379420	313975	368617	360265	307399	412247
FANTOM	$\varepsilon = 0.1$	30605	36608	30809	36242	35682	30223	40543
λ-GREEDY	$\lambda = k$	2709	3080	2895	2895	3080	2709	3080

algorithm	parameter	video (15)	video (16)	video (17)	video (18)	video (19)	video (20)
FANTOM	$\varepsilon = 0.001$	3727241	3321987	3429387	3555884	3375296	3431653
FANTOM	$\varepsilon = 0.01$	374322	330994	345333	354694	336718	343419
FANTOM	$\varepsilon = 0.1$	36073	32377	33357	34741	32370	32187
λ-GREEDY	$\lambda = k$	2895	2895	2895	2895	2709	2709

solution quality for each video clip

algorithm	parameter	video (1)	video (2)	video (3)	video (4)	video (5)	video (6)	video (7)
FANTOM	$\varepsilon = 0.001$	19.3818	15.6143	15.4285	10.6228	13.2393	14.0438	9.4999
FANTOM	$\varepsilon = 0.01$	19.3818	15.6143	15.4285	10.6228	13.2393	12.9851	9.4999
FANTOM	$\varepsilon = 0.1$	16.9083	13.9868	13.9942	9.1811	13.2393	12.9851	9.4999
λ-GREEDY	$\lambda = k$	23.9323	21.8122	22.5406	15.0203	19.4932	18.6267	15.5678

algorithm	parameter	video (8)	video (9)	video (10)	video (11)	video (12)	video (13)	video (14)
FANTOM	$\varepsilon = 0.001$	11.0898	16.2864	10.7798	15.9894	15.5939	12.5897	18.7495
FANTOM	$\varepsilon = 0.01$	11.0898	16.2864	10.7798	15.9894	15.5939	12.5897	18.7495
FANTOM	$\varepsilon = 0.1$	9.5612	16.2864	10.7798	14.4139	14.1122	9.5909	17.3095
λ-GREEDY	$\lambda = k$	18.5727	23.9619	17.6612	23.2229	20.8876	18.1164	25.1342

algorithm	parameter	video (15)	video (16)	video (17)	video (18)	video (19)	video (20)
FANTOM	$\varepsilon = 0.001$	16.3391	11.8452	13.6084	16.9964	13.0314	13.0558
FANTOM	$\varepsilon = 0.01$	16.3391	11.8452	13.6084	16.9964	13.0314	13.0558
FANTOM	$\varepsilon = 0.1$	14.7544	11.8452	12.0878	14.0999	11.5619	11.5385
λ-GREEDY	$\lambda = k$	23.8740	19.4916	19.9461	22.2884	18.6588	19.0673

train the parameters U, W. We consider 20 movies from the Frames Labeled In Cinema dataset [15]. Each movie has 200 frames and 7 generated ground summaries consisting of 15 frames each.

We select a representative set of frames, by maximizing the function $\log\det L$ under additional quality feature constraints, viewed as multiple knapsacks. Hence, this task consists of maximizing a non-monotone submodular function under multiple knapsack constraints. We run the λ-GREEDY and FANTOM algorithms on each instance, until no remaining point in the search space yields improvement on the fitness value, without violating side constraints. We then compare the resulting run time and empirical approximation ratio. Since FANTOM depends on a parameter ε [27], we perform three sets of experiments for $\varepsilon = 0.1$, $\varepsilon = 0.01$, and $\varepsilon = 0.001$. The parameter λ for the λ-GREEDY is always set to $\lambda = k$. We have no indications that a lower λ yields improved solution quality on this set of instances.

Results for the run time and approximation guarantee are displayed in Table 1. We clearly see that the λ-GREEDY outperforms FANTOM in terms of solution quality. Furthermore, the run time of FANTOM is orders of magnitude worse than that of our λ-GREEDY. This is probably due to the fact that the FANTOM requires a very low density threshold to get to a good solution on these instances. The code for this set of experiments is available upon request.

7 DYNAMIC EXPERIMENTS

The aim of these experiments is to show that, when constraints quickly change dynamically, the robust λ-DGREEDY significantly outperforms the λ-GREEDY with a restart policy, that re-sets the optimization process each time new weights are given. To this end, we simulate a setting where updates change dynamically, by introducing controlled posterior noise on the weights. At each update, we run the λ-GREEDY from scratch, and let the λ-DGREEDY continue without a restart policy. We consider two set of dynamic experiments.

The Maximum Entropy Sampling Problem We consider the problem of maximizing the entropy f under a partition matroid con-

Table 2. For the respective set of experiments, Maximum Entropy Sampling (left) and Video Summarization (right), the mean and standard deviation of the solution quality over time within one run of the respective algorithms for different update frequencies [τ] and different dynamic update standard deviations [σ]. With $X^{(+)}$ we denote that the algorithm labelled X significantly outperformed the other one.

		λ-GREEDY (1)		λ-DGREEDY (2)		
τ	σ	mean	sd	mean	sd	stat
10K	0.075	107.4085	1.35	**130.5897**	12.28	$2^{(+)}$
20K	0.075	195.0352	2.99	**213.3886**	11.15	$2^{(+)}$
30K	0.075	263.8143	4.80	**279.7259**	11.38	$2^{(+)}$
40K	0.075	319.6090	6.78	**329.8283**	10.80	$2^{(+)}$
50K	0.075	351.6649	8.47	**353.6060**	9.19	$2^{(+)}$
10K	0.05	107.6904	0.73	**132.7425**	8.79	$2^{(+)}$
20K	0.05	195.5784	1.82	**216.0143**	8.00	$2^{(+)}$
30K	0.05	264.6016	3.13	**282.2722**	7.91	$2^{(+)}$
40K	0.05	320.6179	4.66	**332.0804**	7.57	$2^{(+)}$
50K	0.05	352.8980	6.10	**355.0595**	6.64	$2^{(+)}$
10K	0.10	107.2164	1.56	**124.4393**	14.81	$2^{(+)}$
20K	0.10	194.4608	3.43	**208.3658**	13.35	$2^{(+)}$
30K	0.10	262.7148	5.57	**274.8722**	13.64	$2^{(+)}$
40K	0.10	317.9336	7.96	**325.7493**	12.93	$2^{(+)}$
50K	0.10	349.5698	10.04	**351.1627**	10.91	$2^{(+)}$

		λ-GREEDY (1)		λ-DGREEDY (2)		
τ	σ	mean	sd	mean	sd	stat
10K	0.075	36.550	2.1E-14	**264.256**	69.21	$2^{(+)}$
20K	0.075	69.760	7.2E-14	**269.639**	57.77	$2^{(+)}$
30K	0.075	102.969	4.3E-14	**271.531**	53.20	$2^{(+)}$
40K	0.075	136.956	8.6E-14	**272.210**	51.18	$2^{(+)}$
50K	0.075	174.657	1.30	**272.968**	49.38	$2^{(+)}$
10K	0.05	36.55	2.1E-14	**197.251**	63.49	$2^{(+)}$
20K	0.05	69.760	7.9E-14	**257.641**	52.15	$2^{(+)}$
30K	0.05	102.969	4.3E-14	**259.517**	47.55	$2^{(+)}$
40K	0.05	136.956	8.6E-14	**260.197**	45.46	$2^{(+)}$
50K	0.05	174.840	5.7E-14	**260.955**	43.64	$2^{(+)}$
10K	0.10	36.550	2.1E-14	**269.784**	76.62	$2^{(+)}$
20K	0.10	69.760	7.2E-14	**275.981**	65.69	$2^{(+)}$
30K	0.10	102.969	4.3E-14	**277.891**	61.48	$2^{(+)}$
40K	0.10	136.956	8.6E-14	**278.571**	59.67	$2^{(+)}$
50K	0.10	174.448	2.77	**279.329**	58.05	$2^{(+)}$

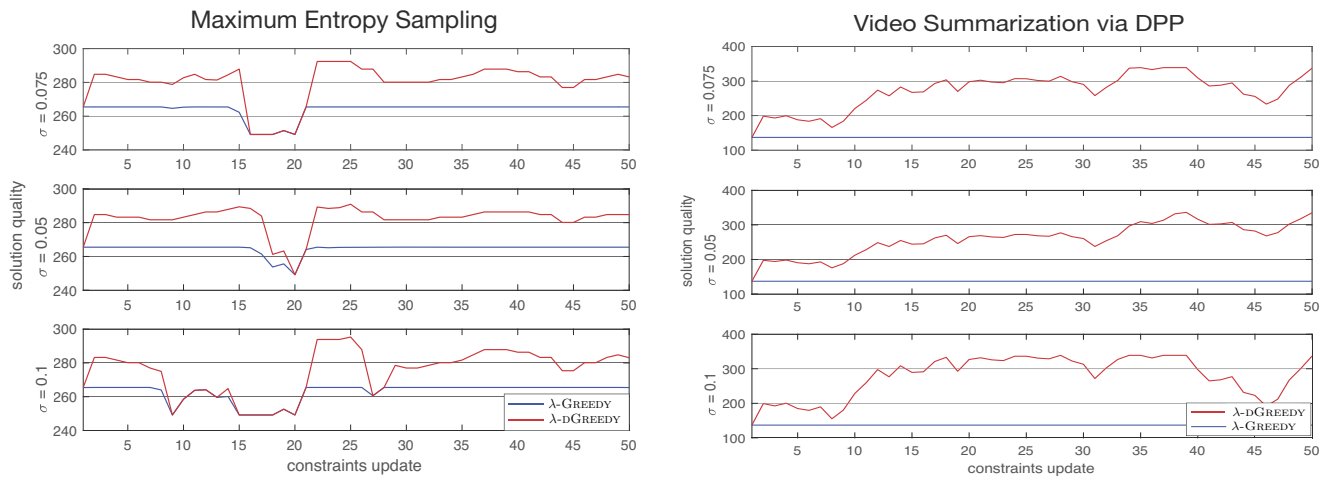

Figure 1. The solution quality achieved by the λ-GREEDY after each constraints' update, using a restart strategy, and the λ-DGREEDY operator. Each plot shows results for a fixed standard deviation choice [σ] and fixed update frequency $\tau = 30K$ (for the Maximum Entropy Sampling) and $\tau = 50K$ (for Video Summarization).

straint. This additional side constraint requires an upper bound on the number of sensors that can be chosen in given geographical areas. Specifically, we partition the total number of time series in seven sets, based on the continent in which the corresponding stations are located. Under this partition set, we then have seven independent cardinality constraints, one for each continent.

We use the Berkeley Earth Surface Temperature Study, which combines 1.6 billion temperature reports from 16 preexisting data archives. This archive contains over 39000 unique stations from around the world. More information on the Berkeley Earth project can be found in [2]. Here, we consider unique time series defined as the average monthly temperature for each station. Taking into account all data between years 2015-2017, we obtain 2736 time series from the corresponding stations. Our experimental framework follows along the lines of [16].

In our dynamic setting, for each continent, a given parameter d_i is defined as a percentage value of the overall number of stations available on that continent, for all $i \in [7]$. We let parameters $d_1, \ldots, d_7$ vary over time, as to simulate a setting where they are updated dynamically. This situation could occur when operational costs slightly vary overtime. We initially set all parameters to use 50% of the available resources, and we introduce a variation of these parameters at regular intervals, according to $\mathcal{N}(0, \sigma^2)$, a Gaussian distribution of mean 0 and variance σ^2, for all $i \in [7]$.

We consider various choices for the standard deviation σ, and various choices for the time span between one dynamic update and the next one (the parameter τ). For each choice of σ and τ, we consider a total of 50 sequences of changes. We perform statistical validation using the Kruskal-Wallis test with 95% confidence. In order to compare the results, we use the Bonferroni post-hoc statistical procedure. This method is used for multiple comparisons of a control algorithm against two or more other algorithms. We refer the reader to [9] for more detailed descriptions of these statistical tests.

We compare the results in terms of the solution quality achieved

at each dynamic update by the λ-GREEDY and the λ-DGREEDY. We summarize our results in the Table 2 (left) as follows. The columns correspond to the results for λ-GREEDY and the λ-DGREEDY respectively, along with the mean value, standard deviation, and statistical comparison. The symbol $X^{(+)}$ is equivalent to the statement that the algorithm labelled as X significantly outperformed the other one.
Table 2 (left) shows that the λ-DGREEDY has a better performance than the λ-GREEDY algorithm with restarts, when dynamic changes occur, especially for the highest frequencies $\tau = 10K, 20K$. This shows that the λ-DGREEDY is suitable in settings when frequent dynamic changes occur. The λ-GREEDY yields improved performance with lower frequencies, but it under-perform the λ-DGREEDY on our dataset.
Figure 1 (left) shows the solution quality values achieved by the λ-GREEDY and the λ-DGREEDY, for different choices of the standard deviation $\sigma = 0.075, 0.05, 0.1$. Again, we observe that the λ-DGREEDY finds solutions that have better quality than the λ-GREEDY with restarts. Even though the λ-DGREEDY in some cases aligns with the λ-GREEDY with restarts, the performance of the λ-DGREEDY is clearly better than that of the simple λ-GREEDY with restarts. The code for this set of experiments is available upon request.

Determinantal Point Processes We conclude with a dynamic set of experiments on a video summarization task as in Section 5. We define the corresponding matrix L using the quality-diversity decomposition, as proposed in [21]. Specifically, we define the coefficients $L_{i,j}$ of this matrix as $L_{i,j} = q(i)k(i,j)q(j)$, with $q(i)$ representing the quality of the i-th frame and $k(i,j)$ being the diversity between the i-th and j-th frame.
For the diversity measure k, we consider commonly used features $f \in F$ for pictures, and we use these features to define corresponding feature vectors v_i^f for each frame i. Then the diversity measure is defined as

$$k(i,j) = \exp\left(-\sum_{f\in F} \frac{||v_i^f - v_j^f||_2^2}{\sigma_f}\right),$$

with σ_f a parameter for this feature[11]. To learn these parameters we use the Markov Chain Monte Carlo (MCMC) method (see [1]).

We use movie clips from the Frames Labeled In Cinema dataset [15]. We use 16 movies with 150-550 frames each to learn the parameters and one test movie with approximately 400 frames for our experiments. For each movie, we generate 5-10 samples (depending on the total amount of frames) of sets with 10-20 frames as training data. We then use MCMC on the training data to learn the parameters for each movie. When testing the λ-GREEDY and the λ-DGREEDY, we use the sample median of the trained parameters.

In this set of experiments, we consider a constraint by which the set of selected frames must not exceed a memory threshold. We define a cost function $c(S)$ as the sum of the size of each frame in S. As each frame comes with its own size in memory, choosing the best frames under certain memory budget is equivalent to maximizing a submodular function under a linear knapsack constraint.
The weight W is given range $[0\%, 100\%]$, with respect to the total weight $c(V)$, and it is updated dynamically throughout the optimization process, according to a Gaussian distribution $\mathcal{N}(0, \sigma^2)$, for a given variance σ^2. This settings simulates a situation by which the overall available memory exhibits small frequent variation.

[11] In our setting we combine the parameters $\sigma_{\text{COLOR2}} = \sigma_{\text{COLOR3}}$ and $\sigma_{\text{SIFT256}} = \sigma_{\text{SIFT512}}$.

We select various parameter choices for the standard deviation σ, and the frequency τ with which a dynamic update occurs. We investigate the settings $\sigma = 0.075, 0.05, 0.1$, and $\tau = 10K, 20K, 30K, 40K, 50K$. Each combination of σ and τ carries out 50 dynamic changes. Again, we validate our results using the Kruskal-Wallis test with 95% confidence. To compare the obtained results, we apply the Bonferroni post-hoc statistical test [9].

The results are presented in the Table 2 (right). We observe that the λ-DGREEDY yields better performance than the λ-GREEDY with restarts when dynamic changes occur. Similar findings are obtained when comparing a different standard deviation choice $\sigma = 0.075$, 0.05, 0.1. Specifically, for the highest frequency $\tau = 10K$, the λ-DGREEDY achieves better results by approximately one order of magnitude.
Figure 1 (right) shows the solution quality values obtained by the λ-DGREEDY and the λ-GREEDY, as the frequency is set to $\tau = 50K$. It can be observed that, for $\sigma = 0.075$, 0.05, 0.1, the λ-DGREEDY significantly outperforms the λ-GREEDY with restarts, for almost all 50 updates. The code for this set of experiments is available upon request.

8 CONCLUSION

Many real-world optimization problems can be approached as submodular maximization with multiple knapsack constraints (see Problem 3). Previous studies for this problem show that it is possible to approach this problem with a variety of heuristics. These heuristics often involve a local search, and require continuous relaxations of the discrete problem, and they are impractical. We propose a simple discrete greedy algorithm (see Algorithm 1) to approach this problem, that has polynomial run time and yields strong approximation guarantees for functions with bounded curvature (see Definition 2 and Theorem 5).

Furthermore, we study the problem of maximizing a submodular function, when knapsack constraints involve dynamic components. We study a setting by which the weights W_i of a given set of knapsack constraints change overtime. To this end, we introduce a robust variation of our λ-GREEDY algorithm that allows for handling dynamic constraints online (see Algorithm 2). We prove that this operator allows to maintain strong approximation guarantees for functions with bounded curvature, when constraints change dynamically (see Theorem 6).

We show that, in static settings, Algorithm 1 competes with FANTOM, which is a popular algorithm for handling these constraints (see Table 1). Furthermore, we show that the λ-DGREEDY is useful in dynamic settings. To this end, we compare the λ-DGREEDY with the λ-GREEDY combined with a restart policy, by which the optimization process starts from scratch at each dynamic update. We observe that the λ-DGREEDY yields significant improvement over a restart in dynamic settings with limited computational time budget (see Figure 1 and Table 2).

Acknowledgement

This work has been supported by the Australian Research Council through grant DP160102401 and by the South Australian Government through the Research Consortium "Unlocking Complex Resources through Lean Processing".

REFERENCES

[1] Raja Hafiz Affandi, Emily B. Fox, Ryan P. Adams, and Benjamin Taskar, 'Learning the parameters of determinantal point process kernels', in *Proc. of ICML*, pp. 1224–1232, (2014).

[2] Berkeley Earth. The berkeley earth surface temperature study. http://www.berkeleyearth.org, 2019.

[3] Andrew An Bian, Joachim M. Buhmann, Andreas Krause, and Sebastian Tschiatschek, 'Guarantees for greedy maximization of non-submodular functions with applications', in *Proc. of ICML*, pp. 498–507, (2017).

[4] Allan Borodin, Aadhar Jain, Hyun Chul Lee, and Yuli Ye, 'Max-sum diversification, monotone submodular functions, and dynamic updates', *ACM Transactions on Algorithms*, **13**(3), 41:1–41:25, (July 2017).

[5] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen, 'Submodular function maximization via the multilinear relaxation and contention resolution schemes', *SIAM Journal of Computing*, **43**(6), 1831–1879, (2014).

[6] Raymond Chiong, Thomas Weise, and Zbigniew Michalewicz, *Variants of evolutionary algorithms for real-world applications*, Springer, 2012.

[7] Jung Jin Cho, Yong Chen, and Yu Ding, 'On the (co)girth of a connected matroid', *Discrete Applied Mathematics*, **155**(18), 2456–2470, (2007).

[8] Michele Conforti and Gérard Cornuéjols, 'Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem', *Discrete Applied Mathematics*, **7**(3), 251–274, (1984).

[9] Gregory W. Corder and Dale I. Foreman, *Nonparametric statistics for non-statisticians: a step-by-step approach*, Wiley, 2009.

[10] Abhimanyu Das and David Kempe, 'Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection', *Journal of Machine Learning Research*, **19**, 3:1–3:34, (2018).

[11] Benjamin Doerr, Carola Doerr, Aneta Neumann, Frank Neumann, and Andrew M. Sutton, 'Optimization of chance-constrained submodular functions', in *Proc. of AAAI*, (2020). to appear.

[12] Vanja Doskoč, Tobias Friedrich, Andreas Gbel, Frank Neumann, Aneta Neumann, and Francesco Quinzan. Non-monotone submodular maximization with multiple knapsacks in static and dynamic settings. https://arxiv.org/abs/1911.06791, 2019.

[13] Salman Fadaei, MohammadAmin Fazli, and MohammadAli Safari, 'Maximizing non-monotone submodular set functions subject to different constraints: Combined algorithms', *Operations Research Letters*, **39**(6), 447 – 451, (2011).

[14] Mahyar Fazlyab, Santiago Paternain, Victor M. Preciado, and Alejandro Ribeiro, 'Interior point method for dynamic constrained optimization in continuous time', in *ACC*, pp. 5612–5618. IEEE, (2016).

[15] FLIC Dataset. Frames labeled in cinema. https://bensapp.github.io/flic-dataset.html, 2019.

[16] Tobias Friedrich, Andreas Göbel, Frank Neumann, Francesco Quinzan, and Ralf Rothenberger, 'Greedy maximization of functions with bounded curvature under partition matroid constraints', in *Proc. of AAAI*, pp. 2272–2279, (2019).

[17] Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha, 'Diverse sequential subset selection for supervised video summarization', in *Proc. of NIPS*, pp. 2069–2077, (2014).

[18] Sanjay Dominik Jena, Jean-François Cordeau, and Bernard Gendron, 'Solving a dynamic facility location problem with partial closing and reopening', *Computers & Operations Research*, **67**(C), 143–154, (March 2016).

[19] Samir Khuller, Anna Moss, and Joseph Naor, 'The budgeted maximum coverage problem', *Information Processing Letters*, **70**(1), 39–45, (1999).

[20] Andreas Krause, Ajit Paul Singh, and Carlos Guestrin, 'Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies', *Journal of Machine Learning Research*, **9**, 235–284, (2008).

[21] Alex Kulesza and Ben Taskar, 'Structured determinantal point processes', in *Proc. of NIPS*, pp. 1171–1179, (2010).

[22] Alex Kulesza and Ben Taskar, 'Determinantal point processes for machine learning', *Foundations and Trends in Machine Learning*, **5**(2-3), 123–286, (2012).

[23] Ariel Kulik, Hadas Shachnai, and Tami Tamir, 'Approximations for monotone and nonmonotone submodular maximization with knapsack constraints', *Mathematics of Operations Research*, **38**(4), 729–739, (2013).

[24] Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko, 'Non-monotone submodular maximization under matroid and knapsack constraints', in *Proc. of STOC*, pp. 323–332, (2009).

[25] Hui Lin and Jeff Bilmes, 'Multi-document summarization via budgeted maximization of submodular functions', in *Proc. of NAACL-HLT*, pp. 912–920, (2010).

[26] Odile Macchi, 'The coincidence approach to stochastic point processes', *Advances in Applied Probability*, **7**(1), 83122, (1975).

[27] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi, 'Fast constrained submodular maximization: Personalized data summarization', in *Proc. of ICML*, pp. 1358–1367, (2016).

[28] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi, 'Fast constrained submodular maximization: Personalized data summarization', in *Proc. of ICML*, pp. 1358–1367, (2016).

[29] Baharan Mirzasoleiman, Stefanie Jegelka, and Andreas Krause, 'Streaming non-monotone submodular maximization: Personalized video summarization on the fly', in *Proc. of AAAI*, pp. 1379–1386, (2018).

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

[31] Vahid Roostapour, Aneta Neumann, and Frank Neumann, 'On the performance of baseline evolutionary algorithms on the dynamic knapsack problem', in *Proc of. PPSN*, pp. 158–169, (2018).

[32] Vahid Roostapour, Aneta Neumann, Frank Neumann, and Tobias Friedrich, 'Pareto optimization for subset selection with dynamic cost constraints', in *Proc. of AAAI*, pp. 2354–2361, (2019).

[33] Maxim Sviridenko, 'A note on maximizing a submodular set function subject to a knapsack constraint', *Operation Research Letters*, **32**(1), 41–43, (2004).

[34] Maxim Sviridenko, Jan Vondrák, and Justin Ward, 'Optimal approximation for submodular and supermodular optimization with bounded curvature', *Mathematics of Operations Research*, **42**(4), (2017).

[35] A. Vardy, 'The intractability of computing the minimum distance of a code', *IEEE Transactions on Information Theory*, **43**(6), 1757–1766, (1997).

[36] Qilian Yu, Easton Li Xu, and Shuguang Cui, 'Streaming algorithms for news and scientific literature recommendation: Monotone submodular maximization with a d -knapsack constraint', *IEEE Access*, **6**, 53736–53747, (2018).

[37] Haifeng Zhang and Yevgeniy Vorobeychik, 'Submodular optimization with routing constraints', in *Proc. of AAAI*, eds., Dale Schuurmans and Michael P. Wellman, pp. 819–826, (2016).

[38] Jinging Zheng, Zhuolin Jiang, Rama Chellappa, and P. Jonathon Phillips, 'Submodular attribute selection for action recognition in video', in *Proc. of NIPS*, pp. 1341–1349, (2014).