Streaming Non-monotone Submodular Maximization: Personalized Video Summarization on the Fly

Baharan Mirzasoleiman
ETH Zurich, Switzerland
baharanm@ethz.ch

Stefanie Jegelka
MIT, United States
stefje@mit.edu

Andreas Krause
ETH Zurich, Switzerland
krausea@ethz.ch

Abstract

The need for real time analysis of rapidly producing data streams (e.g., video and image streams) motivated the design of streaming algorithms that can efficiently extract and summarize useful information from massive data "on the fly". Such problems can often be reduced to maximizing a submodular set function subject to various constraints. While efficient streaming methods have been recently developed for monotone submodular maximization, in a wide range of applications, such as video summarization, the underlying utility function is non-monotone, and there are often various constraints imposed on the optimization problem to consider privacy or personalization. We develop the first efficient single pass streaming algorithm, STREAMING LOCAL SEARCH, that for any streaming monotone submodular maximization algorithm with approximation guarantee α under a collection of independence systems $\mathcal{I}$, provides a constant $1/\big(1+2/\sqrt{\alpha}+1/\alpha+2d(1+\sqrt{\alpha})\big)$ approximation guarantee for maximizing a non-monotone submodular function under the intersection of $\mathcal{I}$ and d knapsack constraints. Our experiments show that for video summarization, our method runs more than 1700 times faster than previous work, while maintaining practically the same performance.

Introduction

Data summarization–the task of efficiently extracting a representative subset of manageable size from a large dataset–has become an important goal in machine learning and information retrieval. Submodular maximization has recently been explored as a natural abstraction for many data summarization tasks, including image summarization (Tschiatschek et al. 2014), scene summarization (Simon, Snavely, and Seitz 2007), document and corpus summarization (Lin and Bilmes 2011), active set selection in non-parametric learning (Mirzasoleiman et al. 2016) and training data compression (Wei, Iyer, and Bilmes 2015). Submodularity is an intuitive notion of diminishing returns, stating that selecting any given element earlier helps more than selecting it later. Given a set of constraints on the desired summary, and a (pre-designed or learned) submodular utility function f that quantifies the representativeness $f(S)$ of a subset S of items, data summarization can be naturally reduced to a constrained submodular optimization problem.

In this paper, we are motivated by applications of *non-monotone* submodular maximization. In particular, we consider video summarization in a streaming setting, where video frames are produced at a fast pace, and we want to keep an updated summary of the video so far, with little or no memory overhead. This has important applications e.g. in surveillance cameras, wearable cameras, and astro video cameras, which generate data at too rapid a pace to efficiently analyze and store it in main memory. The same framework can be applied more generally in many settings where we need to extract a small subset of data from a large stream to train or update a machine learning model. At the same time, various constraints may be imposed by the underlying summarization application. These may range from a simple limit on the size of the summary to more complex restrictions such as focusing on particular individuals or objects, or excluding them from the summary. These requirements often arise in real-world scenarios to consider privacy (e.g. in case of surveillance cameras) or personalization (according to users' interests).

In machine learning, Determinantal Point Processes (DPP) have been proposed as computationally efficient methods for selecting a diverse subset from a ground set of items (Kulesza, Taskar, and others 2012). They have recently shown great success for video summarization (Gong et al. 2014), document summarization (Kulesza, Taskar, and others 2012) and information retrieval (Gillenwater, Kulesza, and Taskar 2012). While finding the most likely configuration (MAP) is NP-hard, the DPP probability is a log-submodular function, and submodular optimization techniques can be used to find a near-optimal solution. In general, the above submodular function is very non-monotone, and we need techniques for maximizing a non-monotone submodular function in the streaming setting. Although efficient streaming methods have been recently developed for maximizing a monotone submodular function f with a variety of constraints, there is no effective streaming solution for non-monotone submodular maximization under general types of constraints.

In this work, we provide STREAMING LOCAL SEARCH, the first single pass streaming algorithm for non-monotone submodular function maximization, subject to the intersection of a collection of independence systems $\mathcal{I}$ and d knapsack constraints. Our approach builds on local search, a widely used technique for maximizing non-monotone submodular functions in a batch mode. Local search, however,

needs multiple passes over the input, and hence does not directly extend to the streaming setting, where we are only allowed to make a single pass over the data. This work provides a general framework within which we can use any streaming monotone submodular maximization algorithm, INDSTREAM, with approximation guarantee α under a collection of independence systems $\mathcal{I}$. For any such monotone algorithm, STREAMING LOCAL SEARCH provides a constant $1/\big(1{+}2/\sqrt{\alpha}{+}1/\alpha + 2d(1{+}\sqrt{\alpha})\big)$ approximation guarantee for maximizing a non-monotone submodular function under the intersection of $\mathcal{I}$ and d knapsack constraints. Furthermore, STREAMING LOCAL SEARCH needs a memory and update time that is larger than INDSTREAM with a factor of $O(\log(k)/\sqrt{\alpha})$, where k is the size of the largest feasible solution. Using parallel computation, the increase in the update time can be reduced to $O(1/\sqrt{\alpha})$, making our approach an appealing solution in real-time scenarios. We show that for video summarization, our algorithm leads to streaming solutions that provide competitive utility when compared with those obtained via centralized methods, at a small fraction of the computational cost, i.e. more than 1700 times faster.

Related Work

Video summarization aims to retain diverse and representative frames according to criteria such as representativeness, diversity, interestingness, or frame importance (Ngo, Ma, and Zhang 2003; Liu and Kender 2006; Lee, Ghosh, and Grauman 2012). This often requires hand-crafting to combine the criteria effectively. Recently, Gong et al. (2014) proposed a supervised subset selection method using DPPs. Despite its superior performance, this method uses an exhaustive search for MAP inference, which makes it inapplicable for producing real-time summaries.

Local search has been widely used for submodular maximization subject to various constraints. This includes the analysis of greedy and local search by Nemhauser, Wolsey, and Fisher (1978) providing a $1/(p{+}1)$ approximation guarantee for monotone submodular maximization under p matroid constraints. For non-monotone submodular maximization, the most recent results include a $(1{+}O(1/\sqrt{p}))p$-approximation subject to a p-system constraints (Feldman, Harshaw, and Karbasi 2017), a $1/5-\varepsilon$ approximation under d knapsack constraints (Lee et al. 2009), and a $(p+1)(2p+2d+1)/p$-approximation for maximizing a general submodular function subject to a p-system and d knapsack constraints (Mirzasoleiman, Badanidiyuru, and Karbasi 2016).

Streaming algorithms for submodular maximization have gained increasing attention for producing online summaries. For monotone submodular maximization, Badanidiyuru et al. (2014) proposed a single pass algorithm with a $1/2{-}\epsilon$ approximation guarantee under a cardinality constraint k, using $O(k\log k/\epsilon)$ memory. Later, Chakrabarti and Kale (2015) provided a $1/4p$ approximation guarantee for the same problem under the intersection of p matroid constraints. However, the required memory increases polylogarithmically with the size of the data. Finally, Chekuri, Gupta, and Quanrud (2015) presented deterministic and randomized algorithms for maximizing monotone and non-monotone submodular functions subject to a broader range of constraints, namely a p-matchoid. For maximizing a monotone submodular function, their proposed method gives a $1/4p$ approximation using $O(k\log k/\epsilon^2)$ memory (k is the size of the largest feasible solution). For non-monotone functions, they provide a deterministic $1/(9p+1)$ approximation using the $1/(p{+}1)$ offline approximation of Nemhauser, Wolsey, and Fisher (1978). Their randomized algorithm provides a $1/(4p{+}1/\tau_p)$ approximation in expectation, where $\tau_p = (1-\varepsilon)(2-o(1))/(ep)$ (Feldman, Naor, and Schwartz 2011) is the offline approximation for maximizing a non-negative submodular function.

Using the monotone streaming algorithm of Chekuri, Gupta, and Quanrud (2015) with $1/4p$ approximation guarantee, our framework provides a $1/(4p+4\sqrt{p}+1)$ approximation for maximizing a non-monotone function under a p-matchoid constraint, which is a significant improvement over the work of Chekuri, Gupta, and Quanrud (2015). Note that any monotone streaming algorithm with approximation guarantee under a set of independence systems $\mathcal{I}$ (including a p-system constraint, once such an algorithm exists) can be integrated into our framework to provide approximations for non-monotone submodular maximization under the same set of independence systems $\mathcal{I}$, *and d knapsack constraints*.

Problem Statement

We consider the problem of summarizing a stream of data by selecting, on the fly, a subset that maximizes a utility function $f : 2^V \to \mathbb{R}_+$. The utility function is defined on 2^V (all subsets of the entire stream V), and for each $S \subseteq V$, $f(S)$ quantifies how well S represents the ground set V. We assume that f is *submodular*, a property that holds for many widely used such utility functions. This means that for any two sets $S \subseteq T \subseteq V$ and any element $e \in V \setminus T$ we have

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

We denote the *marginal gain* of adding an element $e \in V$ to a summary $S \subset V$ by $f_S(e) = f(S \cup \{e\}) - f(S)$. The function f is *monotone* if $f_S(e) \geq 0$ for all $S \subseteq V$. Here, we allow f to be non-monotone. Many data summarization applications can be cast as an instance of constrained submodular maximization under a set $\zeta \subseteq 2^V$ of constraints:

$$S^* = \text{argmax}_{S\in\zeta} f(S).$$

In this work, we consider a collection of independence systems and multiple knapsack constraints. An independence system is a pair $\mathcal{M}^I = (V, \mathcal{I})$ where V is a finite (ground) set, and $\mathcal{I} \subseteq 2^V$ is a family of independent subsets of V satisfying the following two properties. (i) $\emptyset \in \mathcal{I}$, and (ii) for any $A \subseteq B \subseteq V, B \in \mathcal{I}$ implies that $A \in \mathcal{I}$ (hereditary property). A *matroid* $\mathcal{M} = (V, \mathcal{I})$ is an independence system with exchange property: if $A, B \in \mathcal{I}$ and $|B| > |A|$, there is an element $e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$. The maximal independent sets of $\mathcal{M}$ share a common cardinality, called the rank of $\mathcal{M}$. A *uniform* matroid is the family of all subsets of size at most l. In a *partition* matroid, we have a collection of disjoint sets B_i and integers $0 \leq l_i \leq |B_i|$ where a set A is independent if for every index i, we have $|A \cap B_i| \leq l_i$. A *p-matchoid* generalizes matchings and intersection of matroids. For q matroids $\mathcal{M}_\ell = (V_\ell, \mathcal{I}_\ell)$,

$\ell \in [q]$, defined over overlapping ground sets V_ℓ, and for $V = \cup_{\ell=1}^{q} V_\ell$, $\mathcal{I} = \{S \subseteq V : S \cap V_\ell \in \mathcal{I}_\ell \ \forall \ell\}$, we have that $\mathcal{M}^p = (V, \mathcal{I})$ is a p-matchoid if every element $e \in V$ is a member of V_ℓ for at most p indices. Finally, a *p-system* is the most general type of constraint we consider in this paper. It requires that if $A, B \in \mathcal{I}$ are two maximal sets, then $|A| \leq p|B|$. A *knapsack* constraint is defined by a cost function $c : V \to \mathbb{R}_+$. A set $S \subseteq V$ is said to satisfy the knapsack constraint if $c(S) = \sum_{e \in S} c(e) \leq W$. Without loss of generality, we assume $W = 1$ throughout the paper.

The goal in this paper is to maximize a (non-monotone) submodular function f subject to a set of constraints ζ defined by the intersection of a collection of independence systems $\mathcal{I}$, and d knapsacks. In other words, we would like to find a set $S \in \mathcal{I}$ that maximizes f where for each set of knapsack costs $c_i, i \in [d]$, we have $\sum_{e \in S} c_i(e) \leq 1$. We assume that the ground set $V = \{e_1, \cdots, e_n\}$ is received from the stream in some arbitrary order. At each point t in time, the algorithm may maintain a memory $M_t \subset V$ of points, and must be ready to output a feasible solution $S_t \subseteq M_t$, such that $S_t \in \zeta$.

Video Summarization with DPPs

Suppose that we are receiving a stream of video frames, e.g. from a surveillance or a wearable camera, and we wish to select a subset of frames that concisely represents all the diversity contained in the video. Determinantal Point Processes (DPPs) are good tools for modeling diversity in such applications. DPPs (Macchi 1975) are distributions over subsets with a preference for diversity. Formally, a DPP $\mathcal{P}$ on a set of items $V = \{1, 2, ..., N\}$ defines a discrete probability distribution on 2^V, such that the probability of every $S \subseteq V$ is

$$\mathcal{P}(Y = S) = \frac{\det(L_S)}{\det(I + L)}, \tag{1}$$

where L is a positive semidefinite kernel matrix, and $L_S \equiv [L_{ij}]_{i,j \in S}$, is the restriction of L to the entries indexed by elements of S, and I is the $N \times N$ identity matrix. In order to find the most diverse and informative feasible subset, we need to solve the NP-hard problem of finding $\arg\max_{S \in \mathcal{I}} \det(L_S)$ (Ko, Lee, and Queyranne 1995), where $\mathcal{I} \subset 2^V$ is a given family of feasible solutions. However, the logarithm $f(S) = \log\det(L_S)$ is a (non-monotone) submodular function (Kulesza, Taskar, and others 2012), and we can apply submodular maximization techniques.

Various constraints can be imposed while maximizing the above non-monotone submodular function. In its simplest form, we can partition the video into T segments, and define a diversity-reinforcing partition matroid to select at most k frames from each segment. Alternatively, various content-based constraints can be applied, e.g., we can use object recognition to select at most $k_i \geq 0$ frames showing person i, or to find a summary that is focused on a particular person or object. Finally, each frame can be associated with multiple costs, based on qualitative factors such as resolution, contrast, luminance, or the probability that the given frame contains an object. Multiple knapsack constraints, one for each quality factor, can then limit the total costs of the elements of the solution and enable us to produce a summary closer to human-created summaries by filtering uninformative frames.

Streaming algorithm for constrained submodular maximization

In this section, we describe our streaming algorithm for maximizing a non-monotone submodular function subject to the intersection of a collection of independence systems and d knapsack constraints. Our approach builds on local search, a widely used technique for maximizing non-monotone submodular functions. It starts from a candidate solution S and iteratively increases the value of the solution by either including a new element in S or discarding one of the elements of S (Feige, Mirrokni, and Vondrak 2011). Gupta et al. (2010) showed that similar results can be obtained with much lower complexity by using algorithms for *monotone* submodular maximization, which, however, are run multiple times. Despite their effectiveness, these algorithms need multiple passes over the input and do not directly extend to the streaming setting, where we are only allowed to make one pass over the data. In the sequel, we show how local search can be implemented in a single pass in the streaming setting.

STREAMING LOCAL SEARCH for a collection of independence systems

The simple yet crucial observation underlying the approach of Gupta et al. (2010) is the following. The solution obtained by approximation algorithms for monotone submodular functions often satisfy $f(S) \geq \alpha f(S \cup C^*)$, where $1 \geq \alpha > 0$, and C^* is the optimal solution. In the monotone case $f(S \cup C^*) \geq f(C^*)$, and we obtain the desired approximation factor $f(S) \geq \alpha f(C^*)$. However, this does not hold for non-monotone functions. But, if $f(S \cap C^*)$ provides a good fraction of the optimal solution, then we can find a near-optimal solution for *non-monotone* functions even from the result of an algorithm for *monotone* functions, by pruning elements in S using unconstrained maximization. This still retains a feasible set, since the constraints are downward closed. Otherwise, if $f(S \cap C^*) \leq \varepsilon \text{OPT}$, then running another round of the algorithm on the remainder of the ground set will lead to a good solution.

Algorithm 1 STREAMING LOCAL SEARCH for independence systems

Input: $f : 2^V \to \mathbb{R}_+$, a membership oracle for independence systems $\mathcal{I} \subset 2^V$; and a monotone streaming algorithm INDSTREAM with α-approximation under $\mathcal{I}$.
Output: A set $S \subseteq V$ satisfying $S \in \mathcal{I}$.

```
while stream is not empty do
    D_0 ← {e}    ▷ e is the next element from the stream.
    ▷ LOCAL SEARCH iterations
    for i = 1 to ⌈1/√α + 1⌉ do
        ▷ D_i is the discarded set by INDSTREAM_i
        [D_i, S_i]= INDSTREAM_i(D_{i-1})
        S'_i =UNCONSTRAINED-MAX(S_i).
    S = argmax_i{f(S_i), f(S'_i)}
Return S
```

Backed by the above intuition, we aim to build multiple disjoint solutions simultaneously within a single pass over the

data. Let INDSTREAM be a single pass streaming algorithm for monotone submodular maximization under a collection of independence systems, with approximation factor α. Upon receiving a new element from the stream, INDSTREAM can choose (1) to insert it into its memory, (2) to replace one or a subset of elements in the memory by it, or otherwise (3) the element gets discarded forever. The key insight for our approach is that it is possible to build other solutions from the elements discarded by INDSTREAM. Consider a chain of $q = \lceil 1/\sqrt{\alpha}+1 \rceil$ instances of our streaming algorithm, i.e. $\{\text{INDSTREAM}_1, \cdots, \text{INDSTREAM}_q\}$. Any element e received from the stream is first passed to INDSTREAM_1. If INDSTREAM_1 discards e, or adds e to its solution and instead discards a set D_1 of elements from its memory, then we pass the set D_1 of discarded elements on to be processed by INDSTREAM_2. Similarly, if a set of elements D_2 is discarded by INDSTREAM_2, we pass it to INDSTREAM_3, and so on. The elements discarded by the last instance INDSTREAM_q are discarded forever. At any point in time that we want to return the final solution, we run unconstrained submodular maximization (e.g. the algorithm of Buchbinder et al. (2015)) on each solution S_i obtained by INDSTREAM_i to get S'_i, and return the best solution among $\{S_i, S'_i\}$ for $i \in [1, q]$.

Theorem 1. *Let* INDSTREAM *be a single pass streaming algorithm for monotone submodular maximization under a collection of independence systems $\mathcal{I}$ with approximation guarantee α. Alg. 1 returns a set $S \in \mathcal{I}$ with*

$$f(S) \geq \frac{1}{(1+1/\sqrt{\alpha})^2}\text{OPT},$$

using memory $O(M/\sqrt{\alpha})$, and average update time $O(T/\sqrt{\alpha})$ per element, where M and T are the memory and update time of INDSTREAM.

The proof of all the theorems can be found in (Mirzasoleiman, Jegelka, and Krause 2017).

We make Theorem 1 concrete via an example: Chekuri, Gupta, and Quanrud (2015) proposed a $1/4p$-approximation streaming algorithm for monotone submodular maximization under a p-matchoid constraint. Using this algorithm as INDSTREAM in STREAMING LOCAL SEARCH, we obtain:

Corollary 2. *With* STREAMING GREEDY *of Chekuri, Gupta, and Quanrud (2015) as* INDSTREAM, STREAMING LOCAL SEARCH *yields a solution $S \in \mathcal{I}$ with approximation guarantee $1/(1+2\sqrt{p})^2$, using $O(\sqrt{p}k\log(k)/\varepsilon)$ memory and $O(p\sqrt{p}k\log(k)/\varepsilon)$ average update time per element, where $\mathcal{I}$ are the independent sets of a p-matchoid, and k is the size of the largest feasible solution.*

Note that any monotone streaming algorithm with approximation guarantee α under a collection of independence systems $\mathcal{I}$ can be integrated into Alg. 1 to provide approximation guarantees for non-monotone submodular maximization under the same set $\mathcal{I}$ of constraints. For example, as soon as there is a subroutine for monotone streaming submodular maximization under a p-system in the literature, one can use it in Alg. 1 as INDSTREAM, and get the guarantee provided in Theorem 1 for maximizing a non-monotone submodular function under a p-system, in the streaming setting.

Algorithm 2 STREAMING LOCAL SEARCH for independence systems $\mathcal{I}$ and d knapsacks

Input: $f : 2^V \to \mathbb{R}_+$, a membership oracle for independence systems $\mathcal{I} \subset 2^V$; d knapsack-cost functions $c_j : V \to [0,1]$; INDSTREAM; and an upper bound k on the cardinality of the largest feasible solution.

Output: A set $S \subseteq V$ satisfying $S \in \mathcal{I}$ and $c_j(S) \leq 1\ \forall j$.

$m = 0$.
while stream is not empty **do**
  $D_0 \leftarrow \{e\}$ ▷ e is the next element from the stream.
  $m = \max(m, f(e))$, $e_m = \text{argmax}_{e \in V} f(e)$.
  $\gamma = \frac{2 \cdot m}{(1+1/\sqrt{\alpha})(1+1/\sqrt{\alpha}+2d\sqrt{\alpha})}$
  $R = \{\gamma, (1+\epsilon)\gamma, (1+\epsilon)^2\gamma, (1+\epsilon)^3\gamma, \ldots, \gamma k\}$
  for $\rho \in R$ in parallel **do**
    ▷ LOCAL SEARCH
    for $i = 1$ to $\lceil 1/\sqrt{\alpha}+1 \rceil$ **do**
      ▷ picks elements only if $\frac{f_{S_i}(e)}{\sum_{j=1}^{d} c_{je}} \geq \rho$
      $[D_i, S_i] = \text{INDSTREAMDENSITY}_i(D_{i-1}, \rho)$
      ▷ unconstrained submodular maximization
      $S'_i = \text{UNCONSTRAINED-MAX}(S_i)$.
    $S_\rho = \text{argmax}_i\{f(S_i), f(S'_i)\}$
  $S = \text{argmax}_{\rho \in R} f(S_\rho)$
Return $\text{argmax}\{f(S), f(\{e_m\})$

STREAMING LOCAL SEARCH for independence systems and multiple knapsack constraints

To respect multiple knapsack constraints in addition to the collection of independence systems $\mathcal{I}$, we integrate the idea of a density threshold (Sviridenko 2004) into our local search algorithm. We use a (fixed) density threshold ρ to restrict the INDSTREAM algorithm to only pick elements if the function value per unit size of the selected elements is above the given threshold. We call this new algorithm INDSTREAMDENSITY. The threshold should be carefully chosen to be below the value/size ratio of the optimal solution. To do so, we need to know (a good approximation to) the value of the optimal solution OPT. To obtain a rough estimate of OPT, it suffices to know the maximum value $m = \max_{e \in V} f(e)$ of any singleton element: submodularity implies that $m \leq \text{OPT} \leq km$, where k is an upper bound on the cardinality of the largest feasible solution satisfying all constraints. We update the value of the maximum singleton element on the fly (Badanidiyuru et al. 2014), and lazily instantiate the thresholds to $\log(k)/\epsilon$ different possible values $(1+\epsilon)^i \in [\gamma, \gamma k]$, for γ defined in Alg. 2. We show that for at least one of the discretized density thresholds we obtain a good enough solution.

Theorem 3. STREAMING LOCAL SEARCH *(outlined in Alg. 2) guarantees*

$$f(S) \geq \frac{1-\epsilon}{(1+1/\sqrt{\alpha})(1+2d\sqrt{\alpha}+1/\sqrt{\alpha})}\text{OPT},$$

with memory $O(M\log(k)/(\epsilon\sqrt{\alpha}))$, and average update time $O(T\log(k)/(\epsilon\sqrt{\alpha}))$ per element, where k is an upper bound on the size of the largest feasible solution, and M and T are the memory and update time of the INDSTREAM *algorithm.*

Table 1: Performance of various video summarization methods with segment size 10 on YouTube and OVP datasets, measured by F-Score (F), Precision (P), and Recall (R).

		Alg. of (Gong et al. 2014)[centralized]		FANTOM [centralized]		STREAMING LOCAL SEARCH	
		Linear	N. Nets	Linear	N. Nets	Linear	N. Nets
YouTube	F	57.8±0.5	60.3±0.5	57.7±0.5	60.3±0.5	58.3±0.5	59.8±0.5
	P	54.2±0.7	59.4±0.6	54.1±0.5	59.1±0.6	55.2±0.5	58.6±0.6
	R	69.8±0.5	64.9±0.5	70.1±0.5	64.7±0.5	70.1±0.5	64.2±0.5
OVP	F	75.5±0.4	77.7±0.4	75.5±0.3	78.0±0.5	74.6±0.2	75.6±0.5
	P	77.5±0.5	75.0±0.5	77.4±0.3	75.1±0.7	76.7±0.2	71.8±0.7
	R	78.4±0.5	87.2±0.3	78.4±0.3	88.6±0.2	76.5±0.3	86.5±0.2

Corollary 4. *By using* STREAMING GREEDY *of Chekuri, Gupta, and Quanrud (2015), we get that* STREAMING LOCAL SEARCH *has an approximation ratio* $(1+\varepsilon)(1+4p+4\sqrt{p}+d(2+1/\sqrt{p}))$ *with* $O(\sqrt{p}k\log^2(k)/\varepsilon^2)$ *memory and update time* $O(p\sqrt{p}k\log^2(k)/\epsilon^2)$ *per element, where* $\mathcal{I}$ *are the independent sets of the* p*-matchoid constraint, and* k *is the size of the largest feasible solution.*

Beyond the Black-Box. Although the DPP probability in Eq. 1 only depends on the selected subset S, in many applications $f(S)$ may depend on the entire data set V. So far, we have adopted the common assumption that f is given in terms of a value oracle (a black box) that computes $f(S)$. Although in practical settings this assumption might be violated, many objective functions are *additively decomposable* over the ground set V (Mirzasoleiman et al. 2016). That means, $f(S)=\frac{1}{V}\sum_{e\in V} f_e(S)$, where $f_e(S)$ is a non-negative submodular function associated with every data point $e\in V$, and $f_e(.)$ can be evaluated without access to the full set V. For decomposable functions, we can approximate $f(S)$ by $f_W(S)=\frac{1}{W}\sum_{e\in W} f_e(S)$, where W is a uniform sample from the stream (e.g. using reservoir sampling (Vitter 1985)).

Theorem 5 (Badanidiyuru et al. (2014)). *Assume that* f *is decomposable, all of* $f_e(S)$ *are bounded, and w.l.o.g.* $|f_e(S)|\leq 1$. *Let* W *be uniformly sampled from* V. *Then for* $|W|\geq \frac{2k^2\log(2/\delta)+2k^3\log(V)}{\varepsilon^2}$, *we can ensure that with probability* $1-\delta$, STREAMING LOCAL SEARCH *guarantees*

$$f(S)\geq \frac{1-\epsilon}{(1+1/\sqrt{\alpha})(1+2d\sqrt{\alpha}+1/\sqrt{\alpha})}(\text{OPT}-\varepsilon).$$

Experiments

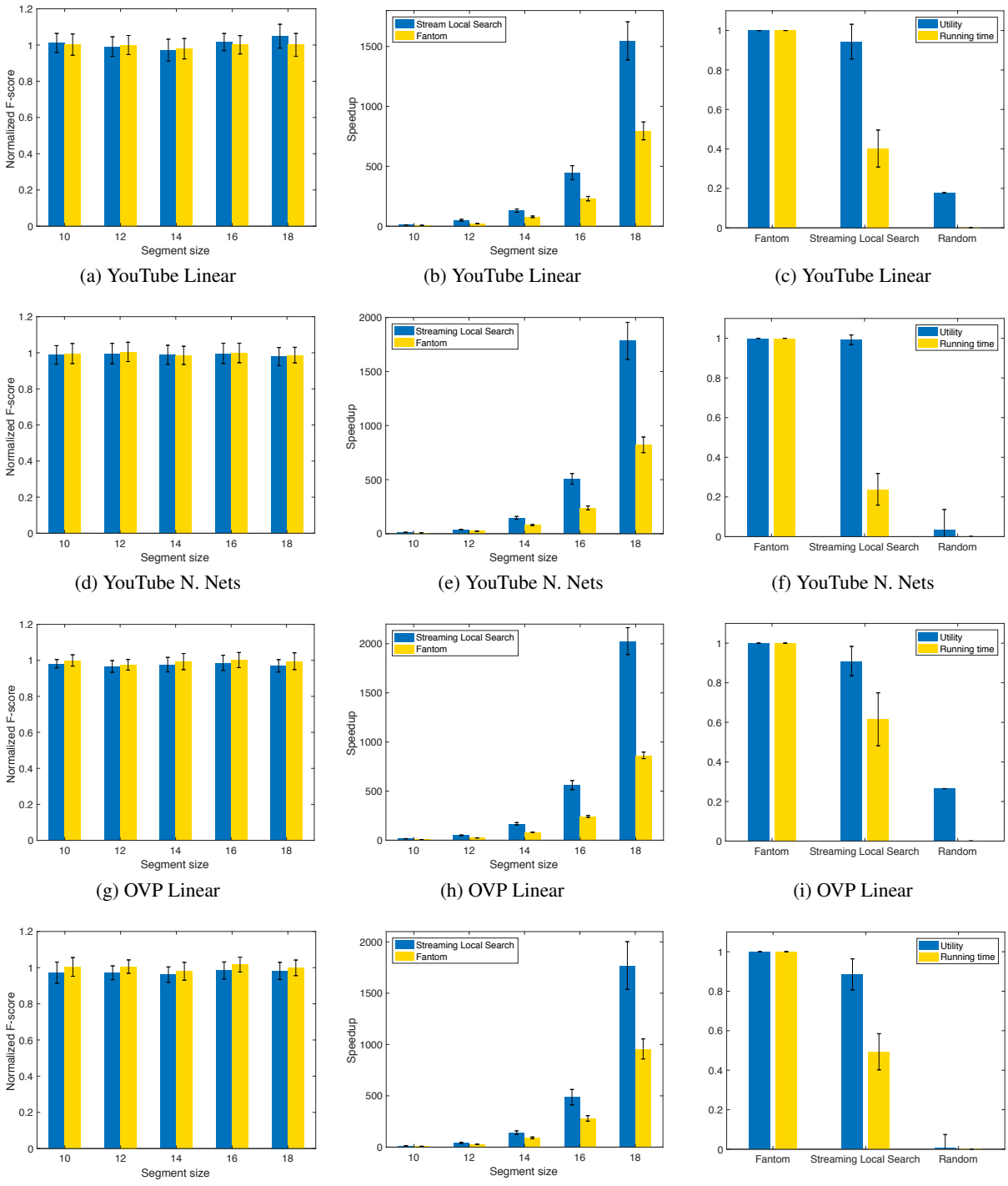
In this section, we apply STREAMING LOCAL SEARCH to video summarization in the streaming setting. The main goal of this section is to validate our theoretical results and demonstrate the effectiveness of our method in practical scenarios, where the existing streaming algorithms are incapable of providing any quality guarantee for the solutions. In particular, for streaming non-monotone submodular maximization under a collection of independence systems and multiple knapsack constraints, none of the previous works provide any theoretical guarantees. We use the streaming algorithm of Chekuri, Gupta, and Quanrud (2015) for monotone submodular maximization under a p-matchoid constraint as INDSTREAM, and compare the performance of our method[1] with exhaustive search (Gong et al. 2014), and a centralized method for maximizing a non-monotone submodular function under a p-system and multiple knapsack constraints, FANTOM (Mirzasoleiman, Badanidiyuru, and Karbasi 2016).

Dataset. For our experiments, we use the Open Video Project (OVP), and the YouTube datasets with 50 and 39 videos, respectively (De Avila et al. 2011). We use the pruned video frames as described in (Gong et al. 2014), where one frame is uniformly sampled per second, and uninformative frames are removed. Each video frame is then associated with a feature vector that consists of Fisher vectors (Perronnin and Dance 2007) computed from SIFT features (Lowe 2004), contextual features, and features computed from the frame saliency map (Rahtu et al. 2010). The size of the feature vectors, v_i, are 861 and 1581 for the OVP and YouTube datasets.

The DPP kernel L (Eq. 1), can be parametrized and learned via maximum likelihood estimation (Gong et al. 2014). For parametrization, we follow (Gong et al. 2014), and use both a linear transformation, i.e. $L_{ij}=v_i^TW^TWv_j$, as well as a non-linear transformation using a one-hidden-layer neural network, i.e. $L_{ij}=z_i^TW^TWz_j$ where $z_i=\tanh(Uv_i)$, and $\tanh(.)$ stands for the hyperbolic transfer function. The parameters, U and W or just W, are learned on 80% of the videos, selected uniformly at random. By the construction of (Gong et al. 2014), we have $\det(L)>0$. However, $\det(L)$ can take values less than 1, and the function is non-monotone. We added a positive constant to the function values to make them non-negative. Following Gong et al. (2014) for evaluation, we treat each of the 5 human-created summaries per video as ground truth for each video.

Sequential DPP. To capture the sequential structure in video data, Gong et al. (2014) proposed a sequential DPP. Here, a long video sequence is partitioned into T disjoint yet consecutive short segments, and for selecting a subset S_t from each segment $t\in[1,T]$, a DPP is imposed over the union of the frames in the segment t and the selected subset S_{t-1} in the immediate past frame $t-1$. The conditional distribution of the selected subset from segment t is thus given by $\mathcal{P}(S_t|S_{t-1})=\frac{\det(L_{S_t\cup S_{t-1}})}{\det(I_t+L_{S_{t-1}\cup V_t})}$, where V_t denotes all the video frames in segment t, and I_t is a diagonal matrix in which the elements corresponding to S_{t-1} are zeros and

[1] Our code is available at github.com/baharanm/non-mon-stream



(a) YouTube Linear (b) YouTube Linear (c) YouTube Linear

(d) YouTube N. Nets (e) YouTube N. Nets (f) YouTube N. Nets

(g) OVP Linear (h) OVP Linear (i) OVP Linear

(j) OVP N. Nets (k) OVP N. Nets (l) OVP N. Nets

Figure 1: Performance of STREAMING LOCAL SEARCH compared to the other benchmarks. a), d) show the ratio of the F-score obtained by STREAMING LOCAL SEARCH and FANTOM vs. the F-score obtained by the method of Gong et al. (2014), using the sequential DPP objective and linear embeddings on YouTube and OVP datasets. g), j) show the relative F-scores for non-linear features from a one-hidden-layer neural network. b), e), h), k) show the speedup of STREAMING LOCAL SEARCH and FANTOM over the method of Gong et al. (2014). c), f), i), l) show the utility and running time for STREAMING LOCAL SEARCH and random selection vs. the utility and running time of FANTOM, using the original DPP objective.

Figure 2: Summary produced by STREAMING LOCAL SEARCH, focused on judges and singer for YouTube video 106.

Figure 3: Summary produced by method of Gong et al. (2014) (top row), vs. STREAMING LOCAL SEARCH (middle row), and a user selected summary (bottom row), for YouTube video 105.

the elements corresponding to S_t are 1. MAP inference for the sequential DPP is as hard as for the standard DPP, but submodular optimization techniques can be used to find approximate solutions. In our experiments, we use a sequential DPP as the utility function in all the algorithms.

Results. Table 1 shows the F-score, Precision and Recall for our algorithm, that of Gong et al. (2014) and FANTOM (Mirzasoleiman, Badanidiyuru, and Karbasi 2016), for segment size $|V_t| = 10$. It can be seen that in all three metrics, the summaries generated by STREAMING LOCAL SEARCH are competitive to the two centralized baselines.

Fig. 1a, 1g show the ratio of the F-score obtained by STREAMING LOCAL SEARCH and FANTOM vs. the F-score obtained by exhaustive search (Gong et al. 2014) for varying segment sizes, using linear embeddings on the YouTube and OVP datasets. It can be observed that our streaming method achieves the same solution quality as the centralized baselines. Fig. 1b, 1h show the speedup of STREAMING LOCAL SEARCH and FANTOM over the method of Gong et al. (2014), for varying segment sizes. We note that both FANTOM and STREAMING LOCAL SEARCH obtain a speedup that is exponential in the segment size. In summary, STREAMING LOCAL SEARCH achieves solution qualities comparable to (Gong et al. 2014), but 1700 times faster than (Gong et al. 2014), and 2 times faster than FANTOM for larger segment size. This makes our streaming method an appealing solution for extracting real-time summaries. In real-world scenarios, video frames are typically generated at such a fast pace that larger segments make sense. Moreover, unlike the centralized baselines that need to first buffer an entire segment, and then produce summaries, our method generates real-time summaries after receiving each video frame. This capability is crucial in privacy-sensitive applications.

Fig. 1d and 1j show similar results for nonlinear representations, where a one-hidden-layer neural network is used to infer a hidden representation for each frame. We make two observations: First, non-linear representations generally improve the solution quality. Second, as before, our streaming algorithm achieves exponential speedup (Fig. 1e, 1k).

Finally, we also compared the three algorithms with a "standard", non-sequential DPP as the utility function, for generating summaries of length 5% of the video length. Again, our method yields competitive performance with a much shorter running time (Fig. 1c, 1f, 1i, 1l).

Using constraints to generate customized summaries. In our second experiment, we show how constraints can be applied to generate customized summaries. We apply STREAMING LOCAL SEARCH to YouTube video 106, which is a part of America's Got Talent series. It features a singer and three judges in the judging panel. Here, we generated two sets of summaries using different constraints. The top row in Fig. 2 shows a summary focused on the judges. Here we considered 3 uniform matroid constraints to limit the number of frames chosen containing each of the judges, i.e., $\mathcal{I} = \{S \subseteq V : |S \cap V_j| \leq l_j\}$, where $V_j \subseteq V$ is the subset of frames containing judge j, and $j \in [1, 3]$; the V_j can overlap. The limits for all the matroid constraints are $l_j = 3$. To produce real-time summaries while receiving the video, we used the Viola-Jones algorithm (Viola and Jones 2004) to detect faces in each frame, and trained a multiclass support vector machine using histograms of oriented gradients (HOG) to recognize different faces. The bottom row in Fig. 2 shows a summary focused on the singer using one matroid constraint.

To further enhance the quality of the summaries, we assigned different weights to the frames based on the probability for each frame to contain objects, using selective

search (Uijlings et al. 2013). By assigning higher cost to the frames that have low probability of containing objects, and by limiting the total cost of the selected elements by a knapsack, we can filter uninformative and blurry frames, and produce a summary closer to human-created summaries. Fig. 3 compares the result of our method, the method of Gong et al. (2014) and a human-created summary.

Conclusion

We have developed the first streaming algorithm, STREAMING LOCAL SEARCH, for maximizing non-monotone submodular functions subject to a collection of independence systems and multiple knapsack constraints. In fact, our work provides a general framework for converting monotone streaming algorithms to non-monotone streaming algorithms for general constrained submodular maximization. We demonstrated its applicability to streaming video summarization with various personalization constraints. Our experimental results show that our method can speed up the summarization task more than 1700 times, while achieving a similar performance as centralized baselines. This makes it a promising approach for many real-time summarization tasks in machine learning and data mining. Indeed, our method applies to any summarization task with a non-monotone (nonnegative) submodular utility function, and a collection of independence systems and multiple knapsack constraints.

Acknowledgments. This research was partially supported by ERC StG 307036, and NSF CAREER 1553284.

References

Badanidiyuru, A.; Mirzasoleiman, B.; Karbasi, A.; and Krause, A. 2014. Streaming submodular maximization: Massive data summarization on the fly. In *KDD*.

Buchbinder, N.; Feldman, M.; Naor, J. S.; and Schwartz, R. 2014. Submodular maximization with cardinality constraints. In *SIAM Journal on Computing*.

Buchbinder, N.; Feldman, M.; Seffi, J.; and Schwartz, R. 2015. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM Journal on Computing* 44(5).

Chakrabarti, A., and Kale, S. 2015. Submodular maximization meets streaming: Matchings, matroids, and more. *Mathematical Programming* 154(1-2).

Chekuri, C.; Gupta, S.; and Quanrud, K. 2015. Streaming algorithms for submodular function maximization. In *ICALP*.

De Avila, S. E. F.; Lopes, A. P. B.; da Luz, A.; and de Albuquerque Araújo, A. 2011. Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method. *Pattern Recognition Letters* 32(1).

Feige, U.; Mirrokni, V. S.; and Vondrak, J. 2011. Maximizing non-monotone submodular functions. *SIAM Journal on Computing* 40(4).

Feldman, M.; Harshaw, C.; and Karbasi, A. 2017. Greed is good: Near-optimal submodular maximization via greedy optimization. *arXiv preprint arXiv:1704.01652*.

Feldman, M.; Naor, J.; and Schwartz, R. 2011. A unified continuous greedy algorithm for submodular maximization. In *FOCS*.

Gillenwater, J.; Kulesza, A.; and Taskar, B. 2012. Discovering diverse and salient threads in document collections. In *EMNLP*.

Gong, B.; Chao, W.-L.; Grauman, K.; and Sha, F. 2014. Diverse sequential subset selection for supervised video summarization. In *NIPS*.

Gupta, A.; Roth, A.; Schoenebeck, G.; and Talwar, K. 2010. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *WINE*.

Ko, C.-W.; Lee, J.; and Queyranne, M. 1995. An exact algorithm for maximum entropy sampling. *Operations Research* 43(4).

Kulesza, A.; Taskar, B.; et al. 2012. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning* 5(2–3).

Lee, J.; Mirrokni, V. S.; Nagarajan, V.; and Sviridenko, M. 2009. Non-monotone submodular maximization under matroid and knapsack constraints. In *STOC*.

Lee, Y. J.; Ghosh, J.; and Grauman, K. 2012. Discovering important people and objects for egocentric video summarization. In *CVPR*.

Lin, H., and Bilmes, J. 2011. A class of submodular functions for document summarization. In *HLT*.

Liu, T., and Kender, J. 2006. Optimization algorithms for the selection of key frame sequences of variable length. In *ECCV*.

Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60(2).

Macchi, O. 1975. The coincidence approach to stochastic point processes. *Advances in Applied Probability* 7(01).

Mirzasoleiman, B.; Badanidiyuru, A.; and Karbasi, A. 2016. Fast constrained submodular maximization: Personalized data summarization. In *ICML*.

Mirzasoleiman, B.; Karbasi, A.; Sarkar, R.; and Krause, A. 2016. Distributed submodular maximization. *Journal of Machine Learning Research* 17(238):1–44.

Mirzasoleiman, B.; Jegelka, S.; and Krause, A. 2017. Streaming non-monotone submodular maximization: Personalized video summarization on the fly. *arXiv preprint arXiv:1706.03583*.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming* 14(1).

Ngo, C.-W.; Ma, Y.-F.; and Zhang, H.-J. 2003. Automatic video summarization by graph modeling. In *ICCV*.

Perronnin, F., and Dance, C. 2007. Fisher kernels on visual vocabularies for image categorization. In *CVPR*.

Rahtu, E.; Kannala, J.; Salo, M.; and Heikkilä, J. 2010. Segmenting salient objects from images and videos. *ECCV*.

Simon, I.; Snavely, N.; and Seitz, S. M. 2007. Scene summarization for online image collections. In *ICCV*.

Sviridenko, M. 2004. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters* 32(1).

Tschiatschek, S.; Iyer, R. K.; Wei, H.; and Bilmes, J. A. 2014. Learning mixtures of submodular functions for image collection summarization. In *NIPS*.

Uijlings, J. R.; Van De Sande, K. E.; Gevers, T.; and Smeulders, A. W. 2013. Selective search for object recognition. *International journal of computer vision* 104(2).

Viola, P., and Jones, M. J. 2004. Robust real-time face detection. *International journal of computer vision* 57(2).

Vitter, J. S. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)* 11(1):37–57.

Wei, K.; Iyer, R.; and Bilmes, J. 2015. Submodularity in data subset selection and active learning. In *ICML*.

Supplementary Materials.

Analysis of STREAMING LOCAL SEARCH

Proof of theorem 1

Proof. Consider a chain of r instances of our streaming algorithm, i.e. $\{\text{INDSTREAM}_1, \cdots, \text{INDSTREAM}_r\}$. For each $i \in [1, r]$, INDSTREAM_i provides an α-approximation guarantee on the ground set V_i of items it has received. Therefore we have:

$$f(S_i) \geq \alpha f(S_i \cup C_i), \tag{2}$$

where $C_i = C^* \cap V_i$ for all $i \in [1, r]$, and C^* is the optimal solution. Moreover, for each i, S'_i is the solution of the unconstrained maximization algorithm on ground set S_i. Therefore, we have:

$$f(S'_i) \geq \beta f(S_i \cap C_i), \tag{3}$$

where β is the approximation guarantee of the unconstrained submodular maximization algorithm (UNCONSTRAINED-MAX).

We now use the following lemma from (Buchbinder et al. 2014) to bound the total value of the solutions provided by the r instances of INDSTREAM.

Lemma 6 (***Lemma 2.2. of*** **(Buchbinder et al. 2014)).** *Let $f' : 2^V \to R$ be submodular. Denote by $A(p)$ a random subset of A where each element appears with probability at most p (not necessarily independently). Then, $\mathbb{E}[f'(A(p))] \geq (1-p)f'(\emptyset)$.*

Let S be a random set which is equal to every one of the sets $\{S_1, \cdots, S_r\}$ with probability $p = 1/r$. For $f' : 2^V \to R$, and $f'(S) = f(S \cup \text{OPT})$, from Lemma 6 we get:

$$\mathbb{E}[f'(S)] = \mathbb{E}[f(S \cup C^*)] = \frac{1}{r}\sum_{i=1}^{r} f(S_i \cup C^*) \overset{\text{Lemma 6}}{\geq} (1-p)f'(\emptyset) = (1 - \frac{1}{r})f(C^*) \tag{4}$$

Also, note that each instance i of INDSTREAM in the chain has processed all the elements of the ground set V except those that are in the solution of the previous instances of INDSTREAM in the chain. As a result, $V_i = V \setminus \cup_{j=1}^{i-1} S_i$, and for every $i \in [1, r]$, we can write:

$$f(C_i) + f(C^* \cap (\cup_{j=1}^{i-1} S_j)) = f(C_i) + f(\cup_{j=1}^{i-1}(C^* \cap S_j)) = f(C^*). \tag{5}$$

Now, using Eq. 4, and via a similar argument as used in (Feldman, Harshaw, and Karbasi 2017), we can write:

$$\begin{aligned}
(r-1)f(C^*) &\leq \sum_{i=1}^{r} f(S_i \cup C^*) && \text{By Eq. 4} \\
&\leq \sum_{i=1}^{r} \Big[f(S_i \cup C_i) + f\big(\cup_{j=1}^{i-1}(C^* \cap S_j)\big) \Big] && \text{By Eq. 5} \qquad (6) \\
&\leq \sum_{i=1}^{r} \Big[f(S_i \cup C_i) + \sum_{j=1}^{i-1} f(C^* \cap S_j) \Big] && (7) \\
&\leq \sum_{i=1}^{r} \Big[\frac{1}{\alpha} f(S_i) + \frac{1}{\beta}\sum_{j=1}^{i-1} f(S'_j) \Big] && \text{By Eq. 2, Eq. 3} \\
&\leq \sum_{i=1}^{r} \Big[\frac{1}{\alpha} f(S) + \frac{1}{\beta}\sum_{j=1}^{i-1} f(S) \Big] && \text{By definition of } S \text{ in Algorithm 1} \\
&= \left(\frac{r}{\alpha} + \frac{r(r-1)}{2\beta} \right) f(S).
\end{aligned}$$

Hence, we get:

$$f(S) \geq \frac{r-1}{r/\alpha + r(r-1)/2\beta} f(C^*) \tag{8}$$

Taking the derivative w.r.t. r, we get that the ratio is maximized for $r = \left\lceil \sqrt{\frac{2\beta}{\alpha}} + 1 \right\rceil$. Plugging this value into Eq. 8, we have:

$$\begin{aligned} f(S) &\geq \frac{1 - \frac{1}{\sqrt{2\beta/\alpha}+1}}{\frac{1}{\alpha} + \frac{\sqrt{2\beta/\alpha}}{2\beta}} f(C^*) \\ &= \frac{\sqrt{2\beta/\alpha}}{(\sqrt{\frac{2\beta}{\alpha}} + 1)(\frac{1}{\alpha} + \frac{\sqrt{2\beta/\alpha}}{2\beta})} f(C^*) \\ &= \frac{\sqrt{2\beta}}{(\sqrt{2\beta} + 1/\sqrt{\alpha})(1/\sqrt{\alpha} + 1/\sqrt{2\beta})} f(C^*) \\ &= \frac{\sqrt{2\beta}}{(1/\sqrt{\alpha} + 1/\sqrt{2\beta})^2} f(C^*) \end{aligned}$$

Using $\beta = 1/2$ from (Buchbinder et al. 2015), we get the desired result:

$$f(S) \geq \frac{1}{(1/\sqrt{\alpha} + 1)^2} f(C^*)$$

Finally, Corollary 2 follows by replacing $\alpha = 1/4p$ from (Chekuri, Gupta, and Quanrud 2015) and $\beta = 1/2$ from (Buchbinder et al. 2015):

$$f(S) \geq \frac{1}{(2\sqrt{p} + 1)^2} f(C^*)$$

□

For calculating the average update time, we consider the worst case scenario, where every element can go through the entire chain of r instances of INDSTREAM at some point during the run of STREAMING LOCAL SEARCH. Here the total running time of the algorithm is $O(nrT)$, where n is the size of the stream, and T is the update time of INDSTREAM. Hence the average update time per element for STREAMING LOCAL SEARCH is $O(nrT/n) = O(rT)$.

Proof of theorem 3

Proof. Here, a (fixed) density threshold ρ is used to restrict the INDSTREAM to only pick elements if $\frac{f_{S_i}(e)}{\sum_{j=1}^{d} c_{je}} \geq \rho$. We first bound the approximation guarantee of this new algorithm INDSTREAMDENSITY, and then use a similar argument as in the proof ot Theorem 1 to provide the guarantee for STREAMING LOCAL SEARCH. Consider an optimal solution C^* and set:

$$\rho^* = \frac{2}{\left(\frac{1}{\sqrt{\alpha}} + \frac{1}{\sqrt{\beta}}\right)\left(\frac{1}{\sqrt{\alpha}} + 2d\sqrt{\alpha} + \frac{1}{\sqrt{\beta}}\right)} f(C^*). \tag{9}$$

By submodularity we know that $m \leq f(C^*) \leq mk$, where k is an upper bound on the cardinality of the largest feasible solution, and m is the maximum value of any singleton element. Hence:

$$\frac{2m}{\left(\frac{1}{\sqrt{\alpha}} + \frac{1}{\sqrt{\beta}}\right)\left(\frac{1}{\sqrt{\alpha}} + 2d\sqrt{\alpha} + \frac{1}{\sqrt{\beta}}\right)} \leq \rho^* \leq \frac{2mk}{\left(\frac{1}{\sqrt{\alpha}} + \frac{1}{\sqrt{\beta}}\right)\left(\frac{1}{\sqrt{\alpha}} + 2d\sqrt{\alpha} + \frac{1}{\sqrt{\beta}}\right)}.$$

Thus there is a run of the algorithm with density threshold $\rho \in R$ such that:

$$\rho \leq \rho^* \leq (1 + \epsilon)\rho. \tag{10}$$

For the run of the algorithm corresponding to ρ, we call the solution of the first instance $\text{INDSTREAMDENSITY}_1$, S_ρ. If $\text{INDSTREAMDENSITY}_1$ terminates by exceeding some knapsack capacity, we know that for one of the knapsacks $j \in [d]$, we have $c_j(S_\rho) > 1$, and hence also $\sum_{j=1}^{d} c_j(S_\rho) > 1$ (W.l.o.g. we assumed the knapsack capacities are 1). On the other hand, the extra density threshold we used for selecting the elements tells us that for any $e \in S_\rho$, we have $\frac{f_{S_\rho}(e)}{\sum_{j=1}^{d} c_{je}} \geq \rho$. I.e., the marginal gain of every element added to the solution S_ρ was greater than or equal to $\rho \sum_{j=1}^{d} c_{je}$. Therefore, we get:

$$f(S_\rho) \geq \sum_{e \in S_\rho} (\rho \sum_{j=1}^{d} c_{je}) > \rho.$$

Note that S_ρ is not a feasible solution, as it exceeds the j-th knapsack capacity. However, the solution before adding the last element e to S_ρ, i.e. $T_\rho = S_\rho - \{e\}$, and the last element itself are both feasible solutions, and by submodularity, the best of them provide us with the value of at least

$$\max\{f(T_\rho), f(\{e_f\})\} \geq \frac{\rho}{2}.$$

On the other hand, if INDSTREAMDENSITY$_1$ terminates without exceeding any knapsack capacity, we divide the elements in $C^* \setminus S_\rho$ into two sets. Let $C^*_{<\rho}$ be the set of elements from C^* which cannot be added to S_ρ because their density is below the threshold, i.e., $\frac{f_{S_\rho}(e)}{\sum_{i=1}^{d} c_{je}} < \rho$ and $C^*_{\geq\rho}$ be the set of elements from C^* which cannot be added to S_ρ due to independence system constraints. For the elements of the optimal solution C^* which cannot be added to S_ρ because their density is below the threshold, we have:

$$f_{S_\rho}(C^*_{<\rho}) \leq \sum_{e \in C_{<\rho}} \rho \sum_{j=1}^{d} c_{je} = \rho \sum_{j=1}^{d} \sum_{e \in C_{<\rho}} c_{je}$$

Since $C_{<\rho}$ is a feasible solution, we know that $\sum_{e \in C_{<\rho}} c_{je} \leq 1$, and therefore:

$$f_{S_\rho}(C^*_{<\rho}) \leq d\rho \leq \rho \sum_{j=1}^{d} \sum_{e \in C_{<\rho}} c_{je} \leq d\rho \leq d\rho^* \tag{11}$$

On the other hand, if the ground set was restricted to elements that pass the density threshold, then S_ρ would be a subset of that ground set, and the approximation guarantee of INDSTREAM$_1$ still holds; hence from Eq. 2 we know that:

$$f(S_\rho) \geq \alpha f(S_\rho \cup C^*_{\geq\rho}),$$

and thus we obtain:

$$f_{S_\rho}(C^*_{\geq\rho}) = f(S_\rho \cup C^*_{\geq\rho}) - f(S_\rho) \leq \big(\frac{1}{\alpha} - 1\big) f(S_\rho). \tag{12}$$

Adding Eq 11 and 12, and using submodularity we get:

$$f(S_\rho \cup C^*) - f(S_\rho) \leq f_{S_\rho}(C^*_{<\rho}) + f_{S_\rho}(C^*_{\geq\rho}) \leq \big(\frac{1}{\alpha} - 1\big) f(S_\rho) + d\rho$$

Therefore,

$$f(S_\rho) \geq \alpha f(S_\rho \cup C^*) - \alpha d\rho. \tag{13}$$

Now, using a similar argument as in the proof of Theorem 1, we have:

$$\begin{aligned}
(r-1) f(C^*) &\leq \sum_{i=1}^{r} f(S_i \cup C^*) && \text{By Eq. 4} \\
&\leq \sum_{i=1}^{r} f(S_i \cup C_i) + \sum_{i=1}^{r} \sum_{j=1}^{i-1} f(C^* \cap S_j) && \text{By Eq. 7} \\
&\leq \frac{1}{\alpha} \sum_{i=1}^{r} [f(S_i) + \alpha d\rho] + \frac{1}{\beta} \sum_{i=1}^{r} \sum_{j=1}^{i-1} f(S'_j) && \text{By Eq. 13} \\
&\leq \frac{1}{\alpha} \sum_{i=1}^{r} [f(S) + \alpha d\rho] + \frac{1}{\beta} \sum_{i=1}^{r} \sum_{j=1}^{i-1} f(S) && \text{By definition of } S \text{ in Algorithm 2} \\
&= \left(\frac{r}{\alpha} + \frac{r(r-1)}{2\beta} \right) f(S) + rd\rho
\end{aligned}$$

□

Hence, we have:

$$f(S) \geq \frac{r-1}{r/\alpha + r(r-1)/2\beta} f(C^*) - \frac{rd\rho}{r/\alpha + r(r-1)/2\beta} f(C^*)$$

From Eq. 10, we know that $\rho \geq (1-\varepsilon)\rho^*$. Using Eq. 9, we get:

$$f(S) \geq \frac{r-1}{r/\alpha + r(r-1)/2\beta} f(C^*) - \frac{\frac{2rd(1-\varepsilon)}{(1/\sqrt{\alpha}+1/\sqrt{\beta})(1/\sqrt{\alpha}+2d\sqrt{\alpha}+1/\sqrt{\beta})}}{r/\alpha + r(r-1)/2\beta} f(C^*)$$

Plugging in $r = \left\lceil \sqrt{\frac{2\beta}{\alpha}} + 1 \right\rceil$ and simplifying, we get the desired result:

$$\begin{aligned} f(S) &\geq \frac{\sqrt{\frac{2\beta}{\alpha}} - \frac{2d\left(\sqrt{\frac{2\beta}{\alpha}}+1\right)(1-\varepsilon)}{\left(\frac{1}{\sqrt{\alpha}}+\frac{1}{\sqrt{\beta}}\right)\left(\frac{1}{\sqrt{\alpha}}+2d\sqrt{\alpha}+\frac{1}{\sqrt{\beta}}\right)}}{\frac{1}{\alpha}\sqrt{\frac{2\beta}{\alpha}} + \frac{2}{\alpha} + \sqrt{\frac{1}{2\beta\alpha}}} f(C^*) \\ &= \frac{\sqrt{2\beta}\left(\frac{1}{\sqrt{\alpha}}+\frac{1}{\sqrt{\beta}}\right)\left(\frac{1}{\sqrt{\alpha}}+2d\sqrt{\alpha}+\frac{1}{\sqrt{\beta}}\right) - 2d(1-\varepsilon)\left(\sqrt{2\beta}+\sqrt{\alpha}\right)}{\left(\frac{\sqrt{2\beta}}{\alpha}+\frac{2}{\sqrt{\alpha}}+\sqrt{\frac{1}{2\beta}}\right)\left(\frac{1}{\sqrt{\alpha}}+\frac{1}{\sqrt{\beta}}\right)\left(\frac{1}{\sqrt{\alpha}}+2d\sqrt{\alpha}+\frac{1}{\sqrt{\beta}}\right)} f(C^*) \\ &\geq \frac{1-\varepsilon}{(1/\sqrt{\alpha}+1/\sqrt{\beta})(1/\sqrt{\alpha}+2d\sqrt{\alpha}+1/\sqrt{\beta})} f(C^*) \end{aligned}$$

For $\beta = 1/2$ from (Buchbinder et al. 2015), we get the desired result:

$$f(S) \geq \frac{1-\epsilon}{(1+1/\sqrt{\alpha})(1+2d\sqrt{\alpha}+1/\sqrt{\alpha})} f(C^*)$$

Corollary 4 follows by replacing $\alpha = 1/4p$ from (Chekuri, Gupta, and Quanrud 2015) and $\beta = 1/2$ from (Buchbinder et al. 2015):

$$f(S) \geq \frac{1-\varepsilon}{1+4p+4\sqrt{p}+d(2+1/\sqrt{p})} f(C^*)$$

The average update time for one run of the algorithm corresponding to a $\rho \in R$ can be calculated as in the proof of Theorem 1. We run the algorithm for $\log(k)/\varepsilon$ different values of ρ, and hence the average update time of STREAMING LOCAL SEARCH per element is $O(rT\log(k)/\varepsilon)$. However, the algorithm can be run in parallel for the $\log(k)/\varepsilon$ values of ρ (line 7 of Algorithm 2), and hence using parallel processing, the average update time per element is $O(rT)$.