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

Baharan Mirzasoleiman  
ETH Zurich

Stefanie Jegelka  
MIT

Andreas Krause  
ETH Zurich

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 $\alpha$ under a collection of independence systems $\mathcal{I}$, provides a constant $1/(1 + 2/\sqrt{\alpha} + 1/\alpha + 2d(1 + \sqrt{\alpha}))$ 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.

1 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 [1], scene summarization [2], document and corpus summarization [3], active set selection in non-parametric learning [4] and training data compression [5]. 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 concerns (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 [6]. They have recently shown great success for video summarization [7], as well as problems like document summarization [6] and information retrieval [8]. 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 solution for non-monotone submodular maximization under general types of constraints in the streaming setting.

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 independent sets $I$ and $d$ knapsack constraints. Our approach builds on local search, a widely used technique for maximizing non-monotone submodular functions in 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 into which we can plug in any streaming monotone submodular maximization algorithm IndStream with approximation guarantee $\alpha$ under a collection of independent sets $I$. For any such monotone algorithm, Streaming Local Search provides a constant $\frac{1}{1+2/\sqrt{\alpha}+1/\alpha+2d(1+\sqrt{\alpha})}$ approximation guarantee for maximizing a non-monotone submodular function under the intersection of $I$ and $d$ knapsack constraints. Furthermore, the memory and update time of Streaming Local Search scales linearly with $O(\log(k)/\sqrt{\alpha})$ compare to IndStream, where $k$ is the size of the largest feasible solutions. 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.

## 2 Related Work

Video summarization aims to retain diverse and representative frames according to criteria such as representativeness, diversity, interestingness, or importance of the frames [9, 10, 11]. This often requires hand-crafting to combine the criteria effectively. Recently, [7] 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 et al. [12] providing a $1/(p+1)$ approximation for monotone submodular maximization under $p$ matroid constraints. Among the most recent results for non-monotone submodular maximization are a $(1+O(1/\sqrt{p}))p$-approximation subject to a $p$-system constraints [13], a $1/5 - \frac{\epsilon}{p}$ approximation under $d$ knapsack constraints [14], and a $(p+1)/(2p + 2d + 1)$-p-approximation for maximizing a general submodular function subject to a $p$-system and $d$ knapsack constraints [15].

Streaming algorithms for submodular maximization have gained increasing attention for producing online summaries from data streams. Recently, Badanidiyuru et al. [16] proposed a single pass streaming algorithm for monotone maximization that yields a $1/2 - \epsilon$ approximation and needs $O(k \log k/\epsilon)$ memory. Chakrabarti and Kale [17] developed a single pass algorithm for monotone functions over intersections of $p$ matroids, achieving a $1/4p$ approximation guarantee. However, the required memory increases polylogarithmically with the size of the data. Finally, Chekuri et al. [18] presented deterministic and randomized algorithms for maximizing monotone and non-monotone submodular functions subject to a broader range of constraints, namely $p$-matchoids. 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 [12] under a $p$-matchoid constraint. Their randomized algorithm provides a $1/(4p + 1/\tau_p)$ approximation in expectation, where $\tau_p = (1-\epsilon)(2-o(1))/(\epsilon p)$ [19] is the approximation guarantee for maximizing a non-negative submodular function in the offline setting.
We assume that $f_marginal gain$.

We denote the $W$.

We consider the problem of summarizing a stream of data by selecting, on the fly, a subset that

This means that for any two sets $f_marginal gain$.

The goal in this paper is to maximize a (non-monotone) submodular function $S$.

A set $A$ is independent if for every index $e$, $A\subseteq B$ implies that $A\in I$ (hereditary property), and (ii) if $A,B\in I$ and $|B|>|A|$, there is an element $e\in B\setminus A$ such that $A\cup \{e\} \in I$. The maximal independent sets of $M$ share a common cardinality, called the rank of $M$. 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 intersections of matroids. For $q$ matroids defined over overlapping ground sets, $M_t(V_t, I_t), \ell \in [q]$, it requires that every element $e\in V$ is a member of $V_t$ 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 $A,B\in I$ are two maximal sets, then $|A|\leq p|B|$. A knapsack constraint is defined by a cost function $c : V \rightarrow \mathbb{R}_+$. 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 $\zeta$ defined by the intersection of a collection of independent sets $I$ and $d$ knapsacks. In other words, we would like to find a set $S \in I$ that maximizes $f$ where for each knapsack $c_i, i \in [d]$, we have $\sum_{e \in S} c_i(e) \leq 1$. We assume that the ground set $V = \{e_1, \ldots, e_n\}$ is received from the stream in some arbitrary order. At each point $t$ in time, the algorithm may maintain a memory $M_t \in V$ of points, and must be ready to output a candidate feasible solution $S_t \subseteq M_t$, such that $S_t \in \zeta$. Upon receiving an element $e_t$ from the stream, the algorithm may elect to 1) insert it into its memory, 2) discard some elements in its memory and accept $e_t$ instead, or 3) discard $e_t$.

### 4 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 [20] are distributions over subsets with a preference for diversity, and have been successfully applied to video summarization [7], as well as problems like document summarization [6] and information retrieval [8]. Formally, a DPP $\mathcal{P}$ on a set of items $V = \{1, 2, \ldots, N\}$ defines a discrete probability distribution on $2^V$, such that the probability of observing subset $S \subseteq V$ is
\[
P(Y = S) = \frac{\det(L_S)}{\det(I + L)}\tag{1}
\]

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 \( \arg \max_{S \subseteq T} \det(L_S) \) \[21\], where \( T \subseteq 2^N \) is a given family of feasible solutions. However, the logarithm \( f(S) = \log \det(L_S) \) is a (non-monotone) submodular function \[6\], and we can apply submodular maximization techniques.

Various constraints can be imposed while maximizing the above non-monotone submodular utility 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 \( 0 \leq k_i \) frames from person \( i \) in the video, 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.

5 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 independent sets and \( d \) knapsack constraints. Our approach builds on local search, which is a powerful and 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 \( S \) \[22\]. Gupta et al. \[23\] 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 a single pass over the data. In the sequel, we show how local search can be implemented in a single pass in the streaming setting.

5.1 Streaming Local Search for a collection of independence systems

The simple yet crucial observation underlying the approach of Gupta et al. \[22\] 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 \( f(S \cap C^*) \leq \epsilon \text{OPT} \), then running another round of the algorithm on the remainder of the ground set will lead to a good solution.

Backed by the above intuition, we aim to build multiple disjoint solutions simultaneously within a single pass over the data. Let \textsc{IndStream} be a single pass streaming algorithm for monotone submodular maximization under a collection of independent sets, with approximation factor \( \alpha \). Upon receiving a new element from the stream, \textsc{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 and cannot be used later by the algorithm. The key insight for our approach is that it is possible to build other solutions from the elements discarded by \textsc{IndStream}. Consider a chain of \( q = \lfloor 1/\sqrt{\epsilon} + 1 \rfloor \) 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 \textsc{IndStream}_1. If \textsc{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 \textsc{IndStream}_2. Similarly, if a set of elements \( D_2 \) is discarded by \textsc{IndStream}_2, we pass them to \textsc{IndStream}_3, and so on. The elements discarded by the last instance \textsc{IndStream}_q are discarded forever. Finally, at any point in time that we want
We make Theorem 5.1 concrete via an example: Chekuri et al [18] proposed a
algorithm for maximizing a monotone submodular function under a collection of
independence systems \( \mathcal{I} \subset 2^V \); and a monotone streaming
algorithm \textsc{IndStream} with \( \alpha \)-approximation under \( \mathcal{I} \).

**Algorithm 1** 

\textsc{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 \textsc{IndStream} with \( \alpha \)-approximation under \( \mathcal{I} \).

**Output:** A set \( S \subseteq V \) satisfying \( S \in \mathcal{I} \).

1: while stream is not empty do
2: \( D_0 \leftarrow \{ e \} \) \hfill \( e \) is the next element from the stream
3: \hfill \( \triangleright \) 
4: \hfill \( \triangleright \text{Local Search} \) iterations
5: for \( i = 1 \) to \( \lceil 1/\sqrt{\alpha} + 1 \rceil \) do
6: \( D_i \) is the discarded set by \textsc{IndStream} \( \mathcal{I} \)
7: \( [D_i, S_i] = \text{IndStream}(D_{i-1}) \)
8: \hfill \( \triangleright \)
9: \( S = \text{argmax}_i \{ f(S_i), f(S_i') \} \)
10: end while
11: Return \( S \)

To return the final solution, we run unconstrained submodular maximization (e.g. the algorithm of
[24]) on each solution \( S_i \) obtained by \textsc{IndStream} \( \mathcal{I} \) to get \( S_i' \), and return the best solution among
\( \{ S_i, S_i' \} \) for \( i \in [1, q] \).

**Theorem 5.1.** Let \textsc{IndStream} be a streaming algorithm for monotone submodular maximization
under a collection of independent sets \( \mathcal{I} \) with approximation guarantee \( \alpha \). Algorithm 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 \textsc{IndStream}, and \( k \) is an upper bound on the size of the largest feasible solution.

We make Theorem 5.1 concrete via an example: Chekuri et al [18] proposed a \( 1/4p \)-approximation
algorithm for maximizing a monotone submodular function under a \( p \)-matchoid constraint in the
streaming setting. Using this algorithm as \textsc{IndStream} in our \textsc{Streaming Local Search}, we obtain the following result:

**Corollary 5.2.** With \textsc{Streaming Greedy} of [18] as \textsc{IndStream}, \textsc{Streaming Local Search}
 yields a solution \( S \in \mathcal{I} \) with approximation guarantee \( 1/(1 + 2/\sqrt{p})^2 \), using \( O(\sqrt{pk} \log(k)/\varepsilon) \) memory and \( O(p \sqrt{pk} \log(k)/\varepsilon) \) 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 \( \alpha \) under a collection of
independence systems \( \mathcal{I} \) can be integrated into Algorithm 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 Algorithm 1 as \textsc{IndStream}, and get the guarantee provided in Theorem 5.1 for maximizing a non-monotone submodular function under a \( p \)-system, in the streaming setting.

### 5.2 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 [25] into our local search algorithm. We use a (fixed)
density threshold \( \rho \) to restrict the \textsc{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
\textsc{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 \( \text{OPT} \). To obtain a rough estimate of \( \text{OPT} \), 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 [16], and lazily instantiate the
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 $\mathcal{I} \subseteq 2^V$; $d$ knapsack-cost functions $c_j : V \to [0, 1]$; 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 $c_j(S) \leq 1 \forall j$.

1: $m = 0$.
2: **while** stream is not empty **do**
3: $D_0 \leftarrow \{e\}$ \quad \triangleright e is the next element from the stream
4: $m = \max(m, f(e))$, $e_m = \arg\max_{e \in V} f(e)$.
5: $\gamma = (1 + 1/\sqrt{\alpha})(1 + 1/\sqrt{\alpha} + 2d/\alpha)$.
6: $R = \{\gamma_i, (1 + \epsilon)^{\gamma_i}(1 + \epsilon)^{2\gamma_i}, (1 + \epsilon)^{3\gamma_i}, \ldots, \gamma k\}$
7: **for** $\rho \in R$ in parallel **do**
8: \quad **do**
9: \quad \quad **for** $i = 1$ to $\lceil 1/\sqrt{\alpha} + 1 \rceil$ **do**
10: \quad \quad \quad \quad \triangleright picks elements only if $\frac{f_S(e)}{\sum_{j=1}^i c_{e_j}} \geq \rho$
11: \quad \quad \quad $|D_i, S_i| = \text{INDSTREAMDENSITY}_i(D_{i-1}, \rho)$
12: \quad \quad \quad \quad \triangleright unconstrained submodular maximization
13: \quad \quad \quad $S'_i = \text{UNCONSTRAINED-MAX}(S_i)$.
14: \quad \quad **end for**
15: \quad $S'_\rho = \arg\max_{\rho \in \rho} \{f(S'_i), f(S'_i)\}$
16: **end for**
17: $S = \arg\max_{\rho \in \rho} f(S'_\rho)$
18: **end while**
19: Return $\arg\max\{f(S), f(e_m)\}$

thresholds to $\log(k)/\epsilon$ different possible values $(1 + \epsilon)\gamma \in [\gamma, \gamma k]$, for $\gamma$ defined in Algorithm 2. We show that for at least one of the discretized density thresholds we obtain a good enough solution.

**Theorem 5.3.** Streaming Local Search (outlined in Alg. 2) has an approximation guarantee $f(S) \geq \frac{1 - \epsilon}{(1 + 1/\sqrt{\alpha})(1 + 2d/\alpha + 1/\sqrt{\alpha})} \text{OPT}$, 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.

**Corollary 5.4.** By using Streaming Greedy of [18], we get that Streaming Local Search has an approximation ratio $(1 + \epsilon)(1 + 4d + 4\sqrt{\beta} + d(2 + 1/\sqrt{\beta}))$ with $O(\sqrt{\beta}k \log^{2}(k)/\epsilon^2)$ memory and update time $O(p \sqrt{\beta}k \log^{2}(k)/\epsilon^2)$ per element, where $I$ are the independent sets of the $p$-matchoid constraint, and $k$ is the size of the largest feasible solution.

## 6 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 guarantee for the quality of the solution. In particular, for streaming non-monotone submodular maximization under a collection of independent sets and multiple knapsack constraints, none of the previous works provide any theoretical guarantees. We use the streaming algorithm of [18] for monotone submodular maximization under a $p$-matchoid constraint as INDSTREAM, and compare the performance of our method with exhaustive search [2], and a centralized method for maximizing a non-monotone submodular function under a $p$-system and multiple knapsack constraints, FANTOM [15].

**Dataset.** For our experiments, we use the Open Video Project (OVP), and the YouTube datasets with 50 and 39 videos, respectively [20]. We use the pruned video frames as described in [7], where
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).

| Algorithm of [7] (centralized) | FANTOM [15] (centralized) | STREAMING LS |
|--------------------------------|---------------------------|--------------|
|                               | Linear | N. Nets | Linear | N. Nets | Linear | N. Nets |
| YouTube                       | F      |        |        |        |        |        |
| 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      |        |        |        |        |        |
| 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 |

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 [27] computed from SIFT features [28], contextual features, and features computed from the frame saliency map [29]. The size of the feature vectors, $v_i$, are 861 and 1581 for the OVP and YouTube dataset, respectively.

The DPP kernel $L$ (c.f Eq.1), can be parametrized and learned via maximum likelihood estimation [7]. For parametrization, we follow [7], and use both a linear transformation, i.e. $L_{ij} = v_i^T W^T W v_j$, as well as a non-linear transformation using a one-hidden-layer neural network, i.e. $L_{ij} = z^T W z_j$ where $z_j = \tanh(U v_j)$, 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 [7], 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 [7] 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, [7] 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 $\mathcal{P}(S_t|S_{t-1}) = \frac{\det(K_{S_t \cup S_{t-1}})}{\det(K_{S_t \cup R_{S_{t-1} \cup V_t}})}$, 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 the elements corresponding to $S_t$ are 1. Intuitively, the sequential DPP only captures the diversity between the frames in segment $t$, and the selected subset $S_{t-1}$ from the immediate past segment $t - 1$. 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 [7] and FANTOM [15], 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. Figures 1a and 1g show the ratio of the F-score obtained by STREAMING LOCAL SEARCH and FANTOM vs. the F-score obtained by exhaustive search [7] 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. Figures 1a and 1g show the speedup of STREAMING LOCAL SEARCH and FANTOM over the method of [7], 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 [7], but 1700 times faster than [7], 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.

Figures 1c and 1f 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 speed up (Fig. 1c).

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 (Figures 1i, 1c, 1l, 1f).

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 the frames (not necessarily non-overlapping) including judge \( j \), and \( j \in [1, 3] \). The limits \( l_j \) for all the matroid constraints are set to 3.

To produce real-time summaries while receiving the video, we used the Viola-Jones algorithm \[30\] 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 \[31\]. Assigning a higher cost to the frames with a low probability of having objects, and having a knapsack constraint that limits the total cost of the elements of the solution, let us filter uninformative and blurry frames, and produce a summary closer to human-created summaries. Figure 3 compares the result obtained by our method and the method of \[7\] with a human-created summary.

7 Conclusion

We have developed the first streaming algorithm, STREAMING LOCAL SEARCH, for maximizing non-monotone submodular functions subject to a collection of independent sets 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 showed that our method is able to speed up the summarization task more than 1700 times, while achieving a similar performance to the centralized baselines. This makes it a promising approach for real-time summarization tasks. Indeed, our method applies to any summarization task with a non-monotone (nonnegative) submodular utility function, and a collection of independent sets and knapsack constraints. Given the importance of submodular optimization to numerous data mining and machine learning applications, we believe our result is an important step towards providing real-time summaries.
References

[1] Sebastian Tschiatschek, Rishabh K Iyer, Haochen Wei, and Jeff A Bilmes. Learning mixtures of submodular functions for image collection summarization. In Advances in neural information processing systems, 2014.

[2] Ian Simon, Noah Snavely, and Steven M Seitz. Scene summarization for online image collections. In IEEE 11th International Conference on Computer Vision (ICCV), 2007.

[3] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, 2011.

[4] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In Advances in Neural Information Processing Systems, 2013.

[5] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In Proceedings of the International Conference on Machine Learning (ICML), 2015.

[6] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. Foundations and Trends® in Machine Learning, 5(2–3), 2012.

[7] Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. In Advances in Neural Information Processing Systems, 2014.

[8] Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Discovering diverse and salient threads in document collections. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2012.

[9] Chong-Wah Ngo, Yu-Fei Ma, and Hong-Jiang Zhang. Automatic video summarization by graph modeling. In Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on. IEEE, 2003.

[10] Tielu Liu and John Kender. Optimization algorithms for the selection of key frame sequences of variable length. Computer Vision—ECCV 2002, 2006.

[11] Yong Jae Lee, Joydeep Ghosh, and Kristen Grauman. Discovering important people and objects for egocentric video summarization. In Computer Vision and Pattern Recognition (CVPR), 2012 Conference on, 2012.

[12] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. Mathematical Programming, 14(1), 1978.

[13] Moran Feldman, Christopher Harshaw, and Amin Karbasi. Greed is good: Near-optimal submodular maximization via greedy optimization. arXiv preprint arXiv:1704.01652, 2017.

[14] Jon Lee, Vahab S Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009.

[15] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In Proceedings of the 33rd International Conference on Machine Learning (ICML), 2016.

[16] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014.

[17] Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: Matchings, matroids, and more. Mathematical Programming, 154(1-2), 2015.

[18] Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In International Colloquium on Automata, Languages, and Programming, 2015.

[19] Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on, pages 570–579. IEEE, 2011.

[20] Odile Macchi. The coincidence approach to stochastic point processes. Advances in Applied Probability, 7(01), 1975.

[21] Chun-Wa Ko, Jon Lee, and Maurice Queyranne. An exact algorithm for maximum entropy sampling. Operations Research, 43(4), 1995.

[22] Uriel Feige, Vahab S Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. SIAM Journal on Computing, 40(4), 2011.
[23] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *International Workshop on Internet and Network Economics*, 2010.

[24] Niv Buchbinder, Moran Feldman, Joseph Seffi, and Roy Schwartz. A tight linear time \((1/2)\)-approximation for unconstrained submodular maximization. *SIAM Journal on Computing*, 44(5), 2015.

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

[26] Sandra Eliza Fontes De Avila, Ana Paula Brandão Lopes, Antonio da Luz, and Arnaldo de Albuquerque Araújo. Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method. *Pattern Recognition Letters*, 32(1), 2011.

[27] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*, 2007.

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

[29] Esa Rahtu, Juho Kannala, Mikko Salo, and Janne Heikkilä. Segmenting salient objects from images and videos. *Computer Vision–ECCV 2010*, 2010.

[30] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2), 2004.

[31] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2), 2013.

[32] Niv Buchbinder, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1433–1452, 2014.
**Supplementary Materials.**

A  Analysis of Streaming Local Search

Proof of theorem 5.1

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

$$f(S_i) \geq \alpha f(S_i \cup C_i),$$ (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:

$$f(S^*_i) \geq \beta f(S_i \cap C_i),$$ (3)

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

We now use the following lemma from [32] to bound the total value of the solutions provided by the $r$ instances of INDSTREAM.

Lemma A.1 (Lemma 2.2. of [32]). Let $f' : 2^V \rightarrow R$ 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 $\{S_1, \ldots, S_r\}$ with probability $p = 1/r$. For $f'' : 2^V \rightarrow R$, and $f''(S) = f(S \cup OPT)$, from Lemma A.1 we get:

$$\mathbb{E}[f''(S)] = \mathbb{E}[f(S \cup C^*)] = \frac{1}{r} \sum_{i=1}^{r} f(S_i \cup C^*) \geq (1-p)f''(\emptyset) = (1-\frac{1}{r})f(C^*)$$ (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, $V_i = V \setminus \bigcup_{j=1}^{i-1} S_i$, and for every $i \in [1, r]$, we can write:

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

Now, using Eq. 4 and a similar argument as used in [13], we can write:

$$(r-1)f(C^*) \leq \sum_{i=1}^{r} f(S_i \cup C^*) \quad \text{By Eq. 4}$$

$$\leq \sum_{i=1}^{r} \left[ f(S_i \cup C_i) + f(\bigcup_{j=1}^{i-1} (C^* \cap S_j)) \right] \quad \text{By Eq. 5}$$

$$\leq \sum_{i=1}^{r} \left[ f(S_i \cup C_i) + \sum_{j=1}^{i-1} f(C^* \cap S_j) \right] $$

$$\leq \sum_{i=1}^{r} \left[ \frac{1}{\alpha} f(S_i) + \frac{1}{\beta} \sum_{j=1}^{i-1} f(S^*_j) \right] \quad \text{By Eq. 2 Eq. 5}$$

$$\leq \sum_{i=1}^{r} \left[ \frac{1}{\alpha} f(S) + \frac{1}{\beta} \sum_{j=1}^{i-1} f(S) \right] \quad \text{By definition of } S \text{ in Algorithm [1]}

$$= \left( \frac{r}{\alpha} + \frac{r(r-1)}{2\beta} \right) f(S).$$
Hence, we get:
\[ f(S) \geq \frac{r - 1}{r/\alpha + r(r - 1)/2\beta} f(C^*) \]  
(8)

Taking the derivative w.r.t. \( r \), we get that the ratio is maximized for \( r = \left\lceil \sqrt{2/\alpha} + 1 \right\rceil \). Plugging this value into Eq. 8, we have:
\[
 f(S) \geq 1 - \frac{1}{(\sqrt{\frac{2}{\alpha}} + 1)^2} f(C^*)
\]

Using \( \beta = 1/2 \) from [24], we get the desired result:
\[
 f(S) \geq \frac{1}{(1/\sqrt{\alpha} + 1)^2} f(C^*)
\]

Finally, Corollary 5.2 follows by replacing \( \alpha = 1/4\rho \) from [18] and \( \beta = 1/2 \) from [24]:
\[
 f(S) \geq \frac{1}{(2/\sqrt{\beta} + 1)^2} f(C^*)
\]

Proof of theorem 5.3

Proof. Here, a (fixed) density threshold \( \rho \) is used to restrict the IndStream to only pick elements if \( \frac{f_{S}(e)}{\sum_{j=1}^{k} c_{i}} \geq \rho \). We first bound the approximation guarantee of this new algorithm IndStream-Density, and then use a similar argument as in the proof of Theorem 5.1 to provide the guarantee for Streaming Local Search. Consider an optimal solution \( C^* \) and set:
\[
 \rho^* = \frac{2}{\left(\frac{1}{\sqrt{\alpha}} + \frac{1}{\sqrt{\beta}}\right) \left(\frac{1}{\sqrt{\alpha}} + 2d\sqrt{\alpha} + \frac{1}{\sqrt{\beta}}\right)} f(C^*).
\]
(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:
\[
 \rho \leq \rho^* \leq (1 + \epsilon)\rho.
\]
(10)
For the run of the algorithm corresponding to \( \rho \), we call the solution of the first instance \text{IndStreamDensity}_1, \( S_\rho \). 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 \( \frac{f_{\rho}(e)}{\sum_{j=1}^d c_{je}} \geq \rho \). I.e., the marginal gain of every element added to the solution \( S_\rho \) 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 \( S_\rho \) is not a feasible solution, as it exceeds the \( j \)-th knapsack capacity. However, the solution before adding the last element \( e \) to \( S_\rho \), i.e. \( T_\rho = S_\rho \setminus \{e\} \), and the last element itself are both feasible solutions, and by submodularity, the best of them provide us with the value of at least

\[
\max\{f(T_\rho), f(\{e\})\} \geq \frac{\rho}{2}.
\]

On the other hand, if \text{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 \( S_\rho \) because their density is below the threshold, i.e., \( \sum_{j=1}^d c_{je} < \rho \) and \( C_{\geq \rho} \) be the set of elements from \( C^* \) which cannot be added to \( S_\rho \) due to independence system constraints. For the elements of the optimal solution \( C^* \) which cannot be added to \( S_\rho \), because their density is below the threshold, we have:

\[
f(S_\rho) \leq \sum_{e \in C_{<\rho}} (\rho \sum_{j=1}^d c_{je}) = \rho \sum_{j=1}^d \sum_{e \in C_{<\rho}} c_{je}.
\]

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

\[
f(S_\rho) \leq \rho \sum_{j=1}^d \sum_{e \in C_{<\rho}} c_{je} \leq \rho \sum_{j=1}^d \sum_{e \in C_{\geq \rho}} c_{je} \leq \rho d \rho \leq \rho d^*\tag{11}
\]

On the other hand, if the ground set was restricted to elements that pass the density threshold, then \( S_\rho \) would be a subset of that ground set, and the approximation guarantee of \text{IndStream}_1 still holds; hence from Eq.\( \text{2} \) we know that:

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

and thus we obtain:

\[
f(S_\rho)(C^*_{\geq \rho}) = f(S_\rho \cup C^*_{\geq \rho}) - f(S_\rho) \leq \left( \frac{1}{\alpha} - 1 \right) f(S_\rho).\tag{12}
\]

Adding Eq.\( \text{11} \) and \( \text{12} \) and using submodularity we get:

\[
f(S_\rho \cup C^*) - f(S_\rho) \leq f(S_\rho)(C^*_{\geq \rho}) + f(S_\rho)(C^*_{<\rho}) \leq \left( \frac{1}{\alpha} - 1 \right) f(S_\rho) + d \rho
\]

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 5.1 we have:

\[(r - 1)f(C^*) \leq \sum_{i=1}^{r} f(S_i \cup C^*)\]  
By Eq. 4

\[\leq \sum_{i=1}^{r} f(S_i \cup C_i) + \sum_{i=1}^{r} \sum_{j=1}^{i-1} f(C^* \cap S_j)\]  
By Eq. 4

\[\leq \frac{1}{\alpha} \sum_{i=1}^{r} [f(S_i) + \alpha dp] + \frac{1}{\beta} \sum_{i=1}^{r} \sum_{j=1}^{i-1} f(S_j')\]  
By Eq. 13

\[\leq \frac{1}{\alpha} \sum_{i=1}^{r} [f(S) + \alpha dp] + \frac{1}{\beta} \sum_{i=1}^{r} \sum_{j=1}^{i-1} f(S)\]  
By definition of S in Algorithm 2

\[= \left(\frac{r}{\alpha} + \frac{r(r - 1)}{2\beta}\right) f(S) + rd\rho\]  

\[\square\]

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 \(\rho \geq (1 - \varepsilon)\rho^\star\). Using Eq. 9 we get:

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

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

\[f(S) \geq \frac{\sqrt{\frac{2\beta}{\alpha}} - \frac{2d(\sqrt{\frac{2\beta}{\alpha}} + 1)(1 - \varepsilon)}{\alpha \sqrt{\frac{2\beta}{\alpha}} + 2\alpha + \sqrt{\frac{1}{2\beta}}}}{f(C^*)}\]

\[\geq \frac{\sqrt{2\beta} \left(\frac{1}{\sqrt{\alpha}} + \frac{1}{\sqrt{\beta}}\right)^2 \left(\frac{1}{\sqrt{\alpha}} + 2d\sqrt{\alpha} + \frac{1}{\sqrt{\alpha}}\right)}{\left(\frac{1}{\sqrt{\alpha}} + \frac{2\sqrt{\alpha} + 1}{\sqrt{\alpha}}\right) \left(\frac{1}{\sqrt{\alpha}} + \frac{1}{\beta}\right) \left(\frac{1}{\sqrt{\alpha}} + 2d\sqrt{\alpha} + \frac{1}{\sqrt{\alpha}}\right)}\]

For \(\beta = 1/2\) from [24], we get the desired result:

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

Corollary 5.4 follows by replacing \(\alpha = 1/4p\) from [18] and \(\beta = 1/2\) from [24]:

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

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 5.1. We run the algorithm for \(\log(k)/\varepsilon\) different values of \(\rho\), 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 \(\rho\) (line 7 of Algorithm 2), and hence using parallel processing, the average update time per element is \(O(rT)\).