1. Motivation: Streaming Submodular Maximization

- **Submodular maximization**: many applications in data summarization, web mining and social network analysis.
- **Submodularity** is an intuitive diminishing returns property: for a discrete set function $f: 2^V \rightarrow \mathbb{R}_+$, $f$ is **submodular** if for all sets $A \subseteq B \subseteq V$ and $x \not\in B$, we have $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$.
- Submodular function $f$ is **monotone** if for all $A \subseteq B$, $f(A) \leq f(B)$.
- Many practical scenarios where we need to use streaming algorithms:
  - the data arrives at a very fast pace
  - there is only time to read the data once
  - random access to the entire data is not possible

2. Problem Formulation

- Given a non-negative monotone submodular function $f$, find the set $S^*$ of size at most $k$ that maximizes the function $f$:
  $$S^* = \arg\max_{S \subseteq V, |S| \leq k} f(S)$$
- We define $OPT = f(S^*)$.
- It is known that any streaming algorithm with a memory $o(n/k)$ cannot provide an approximation guarantee better than $1/2$.
- **Question 1**: Is there a streaming algorithm with an approximation factor arbitrarily close to $1/2$ whose memory complexity is $O(k)$? **Yes!**
- **Question 2**: Is there a streaming algorithm with a near-optimal adaptive complexity? **Yes!**
- **Question 3**: Can we generalize our algorithms to multi-source data streams? **Yes!**

3. The SieveStreaming++ Algorithm

SieveStreaming++: a streaming algorithm with the optimal approximation factor and memory complexity.

- Choosing elements with marginal gain at least $\tau^* = \frac{OPT}{2}$ from a data stream returns a set $S$ with an objective value of at least $f(S) \geq \frac{OPT}{2}$.
- It is possible to efficiently guess $\tau^*$.

**Algorithm 4. Theoretical Guarantees of SieveStreaming++**

**Theorem**

For a non-negative monotone submodular function $f$ with a cardinality constraint $k$, SieveStreaming++ returns a solution $S$ such that:
- $f(S) \geq (1/2 - \epsilon) \cdot OPT$
- memory complexity is $O(k/\epsilon)$
- number of queries is $O(\log(k)/\epsilon)$ per each element.

**Compare to the state-of-the-art algorithms**:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Ratio</th>
<th>Memory</th>
<th>#Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREEMPTSTREAMING [Buchbinder et al. (2015)]</td>
<td>$1/2$</td>
<td>$O(k)$</td>
<td>$O(k)$</td>
</tr>
<tr>
<td>SIEVESTREAMING [Badanidiyuru et al. (2014)]</td>
<td>$1/2 - \epsilon \cdot O(k \log(k)/\epsilon)$</td>
<td>$O(k \log(k)/\epsilon)$</td>
<td></td>
</tr>
<tr>
<td>SieveStreaming++ (ours)</td>
<td>$1/2 - \epsilon \cdot O((k/\epsilon)^2)$</td>
<td>$O((k/\epsilon)^2)$</td>
<td></td>
</tr>
</tbody>
</table>

**Algorithm 5. An Algorithm with Low Adaptive Complexity**

There is a dire need to implement streaming algorithms with low adaptivity.

- **Batch-Sieve-Streaming++**: first buffer a fraction of the data stream and then, through a parallel threshold filtering procedure, reduce the adaptive complexity, thus substantially lower the running time.

**Theoretical Guarantees of Batch-Sieve-Streaming++**

For a non-negative monotone submodular function $f$ subject to a cardinality constraint $k$, define $N$ to be the total number of elements in the stream and $B$ to be the buffer size. For Batch-Sieve-Streaming++ algorithm we have:
- the approximation factor is $1/2 - \epsilon$
- the memory complexity is $O(B + k/\epsilon)$
- the expected adaptive complexity is $O(N \log(k) \log(k)/\epsilon)$

6. Multi-Source Data Streams

- Often multiple streams co-exist at the same time:
  - it is essential to keep the communication cost low
  - Batch-Sieve-Streaming++ can be generalized to the multi-source scenario with both low adaptivity and low communication cost.

7. Experimental Evaluations

- The goal is to select representative frames from YouTube videos.
- Each frame $f$ is mapped into a feature vector $v_f$.
- Similarity between frames: matrix $M$ where $M_{ij} = e^{-dist(v_f, v_j)}$.
- For a set $S$ the non-negative monotone submodular function $f$ measures the diversity of the vectors in $S$: $f(S) = \log \det(I + \alpha M_S)$.
- The memory and utility of various single-source streaming algorithms.
- The adaptivity of various multi-source streaming algorithms.
- Trade-off between communication cost and adaptivity.