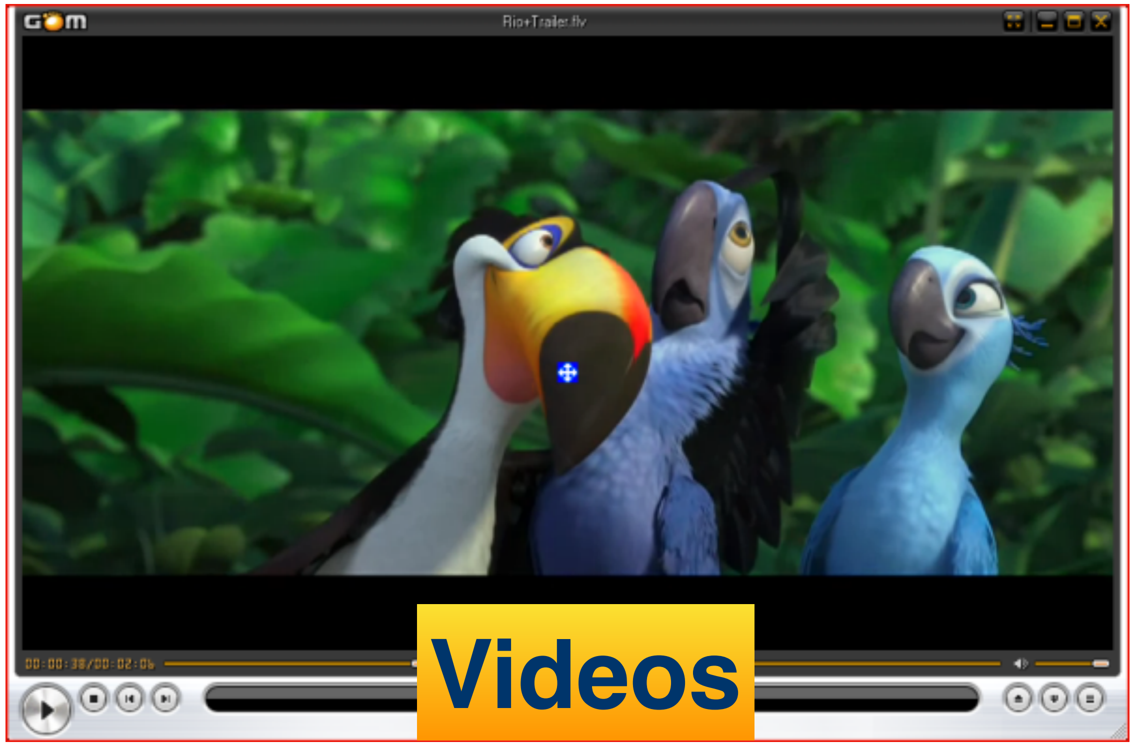
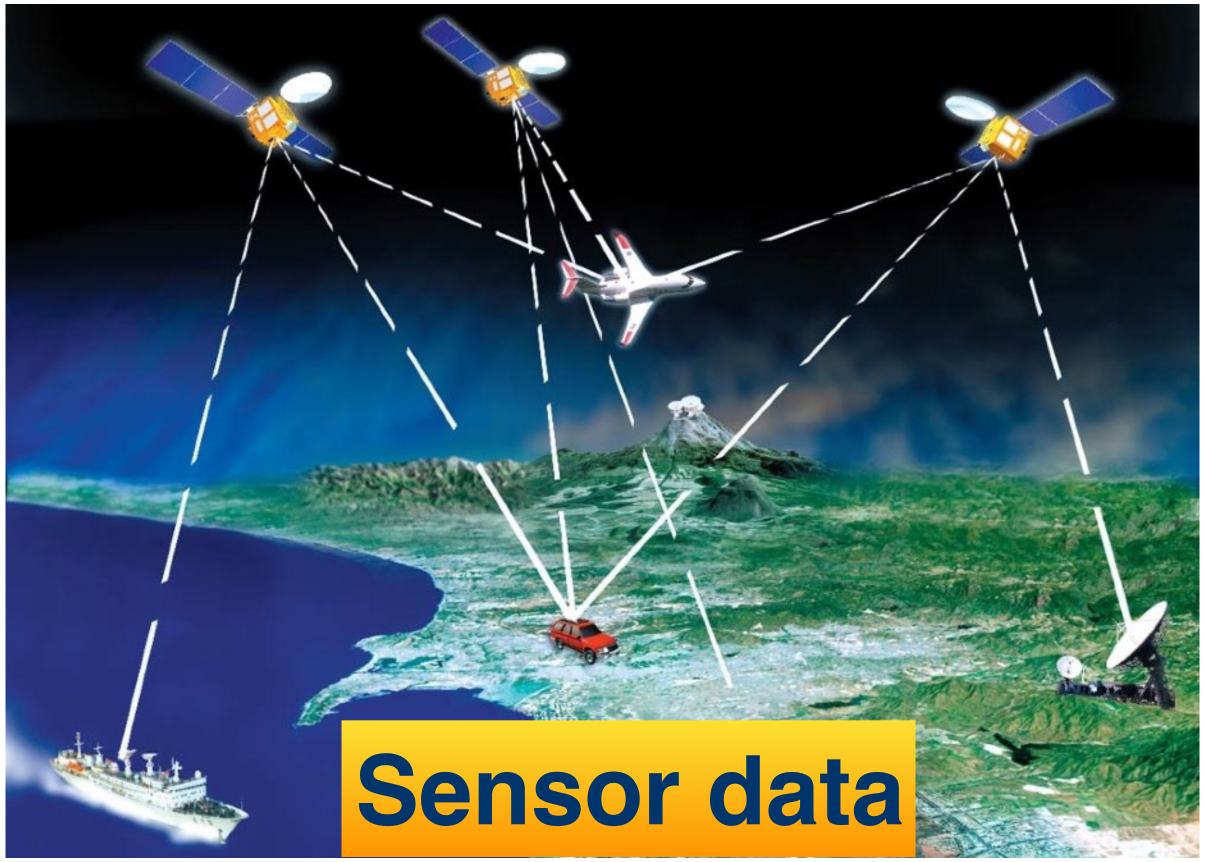
# Do Less, Get More: Streaming Submodular Maximization with Subsampling

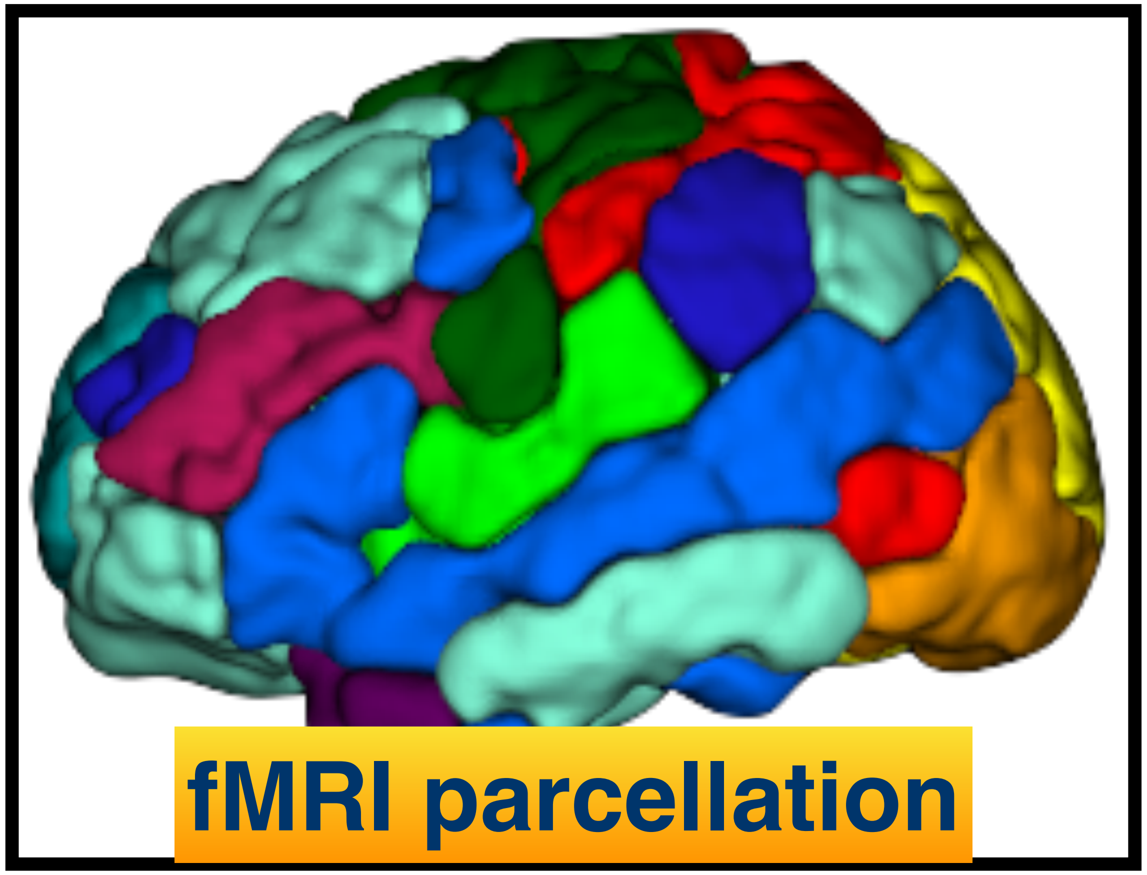
Moran Feldman[1] Amin Karbasi[2] **Ehsan Kazemi**[2]

[1]Open University of Israel and [2]Yale University

# Data Summarization

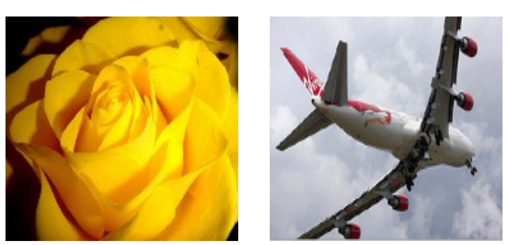
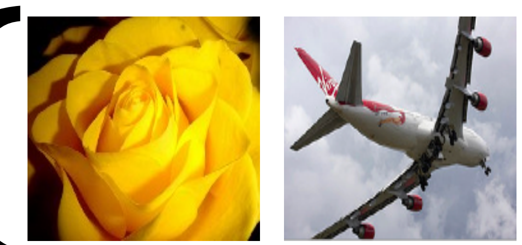



# Submodularity

- Diminishing returns property for set functions.

$V = \{$  $\}$

$f(\{$  $\}) - f(\{$  $\}) \geq$

$f(\{$  $\}) - f(\{$  $\})$

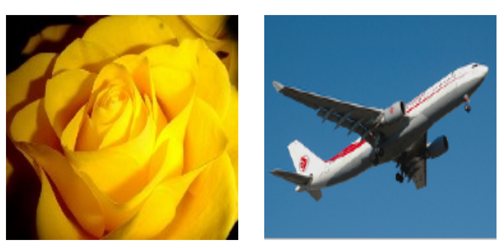
$\forall\ A \subseteq B \subseteq V$ and $x \notin B$

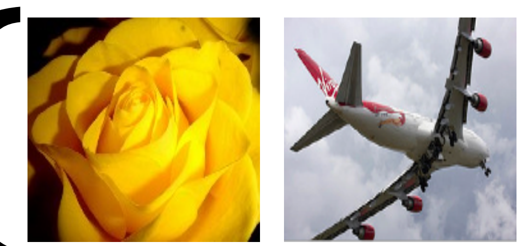
$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$

# Submodularity

- Diminishing returns property for set functions.

$V = \{$  $\}$

$f\big(\{$  $\}\big) - f\big(\{$  $\}\big) \geq$

$f\big(\{$  $\}\big) - f\big(\{$  $\}\big)$

$\forall\ A \subseteq B \subseteq V$ and $x \notin B$

$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$

# Streaming Algorithms

- Many practical scenarios 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** and only a small fraction of the data can be loaded to the main memory

Surveillance camera

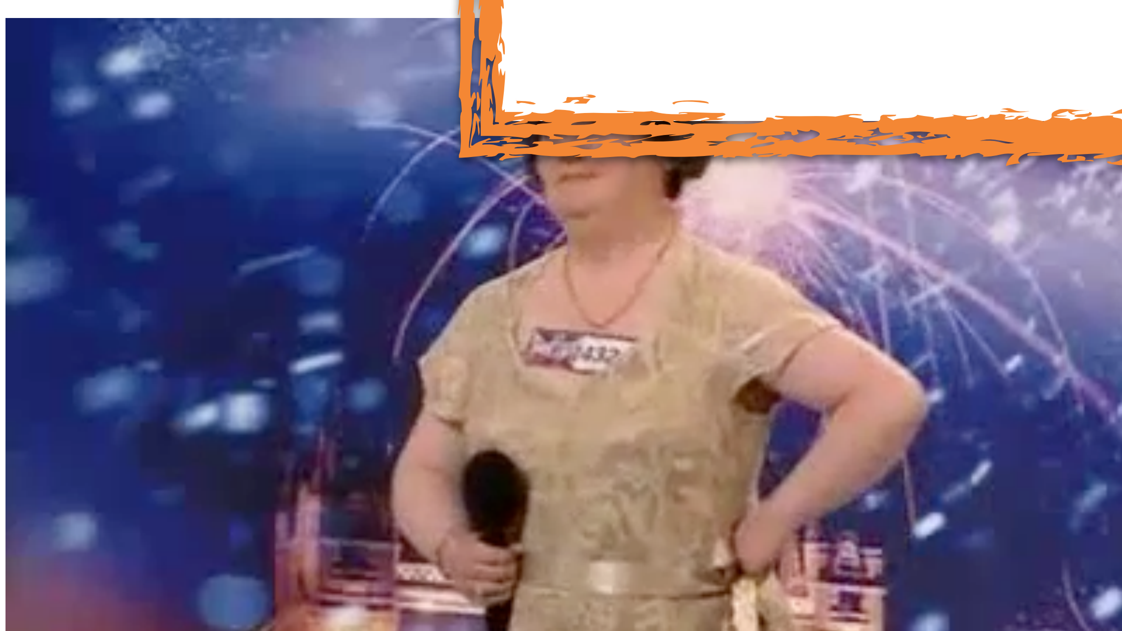
Summary

# Streaming Algorithms

- Many practical scenarios we need to use streaming algorithms:
  - the
  - the
  - **ran** ... **le** and only ... to the mai

**Key challenge:**

Extract small, representative subset out of a massive stream of data



Surveillance camera

Summary

# Constrained Non-Monotone Submodular Maximization

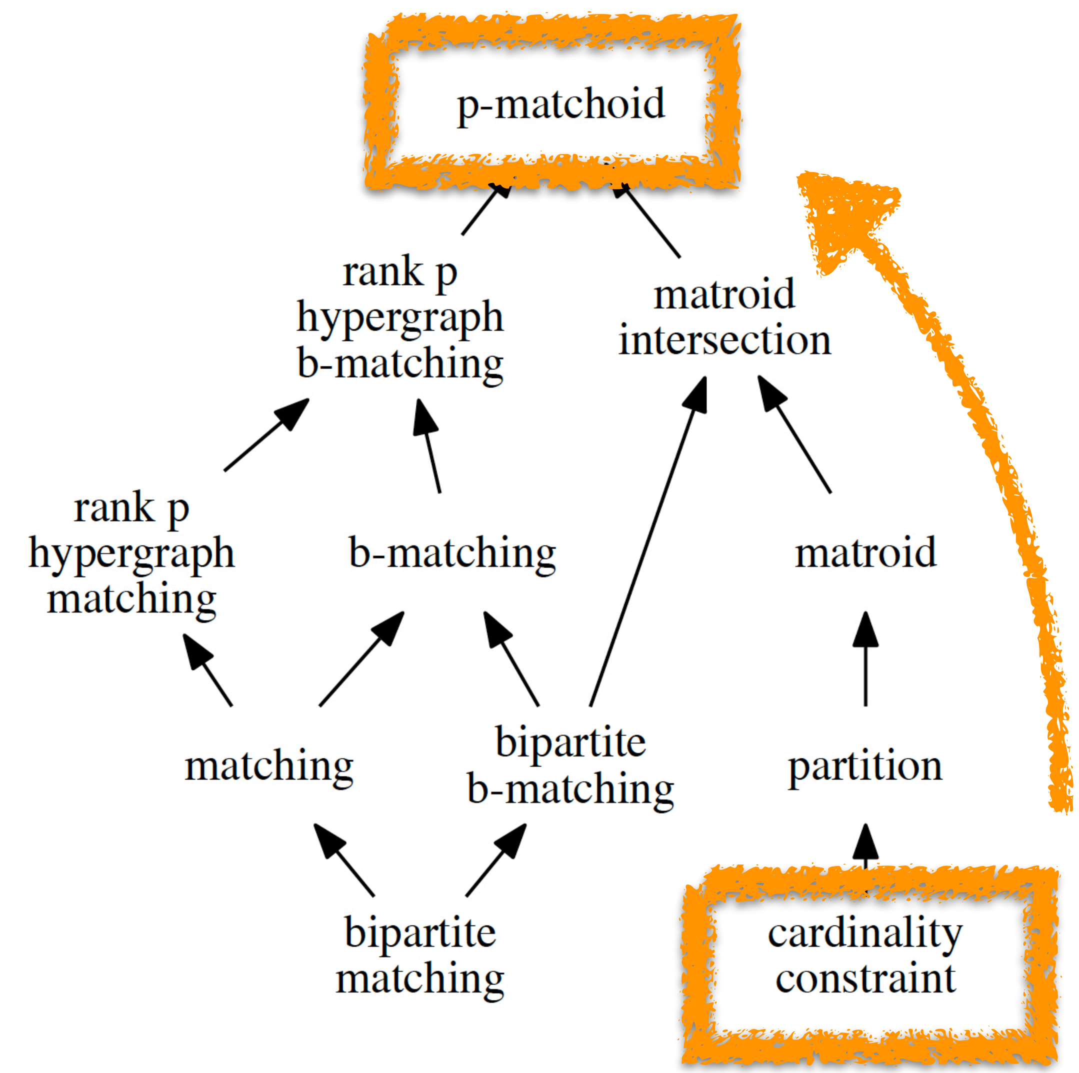
$$S^* = \arg\max_{S \in \mathcal{I}} f(S)$$

constraints

- **Set system**: a pair $(\mathcal{N},\mathcal{I})$, where $\mathcal{N}$ is the ground set and $\mathcal{I} \subseteq 2^{\mathcal{N}}$ is the set of independent sets

- ***p*-matchoid**: a set system $(\mathcal{N},\mathcal{I})$ where there exist *m* matroids $(\mathcal{N}_i,\mathcal{I}_i)$ such that every element of $\mathcal{N}$ appears in the ground set of at most *p* matroids and

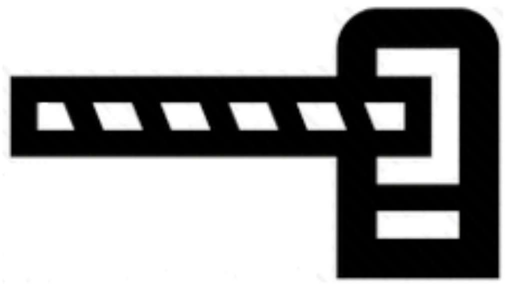
$$\mathcal{I} = \{S \subseteq 2^{\mathcal{N}} \mid \forall_{1 \le i \le m}\ S \cap \mathcal{N}_i \in \mathcal{I}_i\}$$




[Chekuri et al., 2015]

# The Sample-Streaming Algorithm

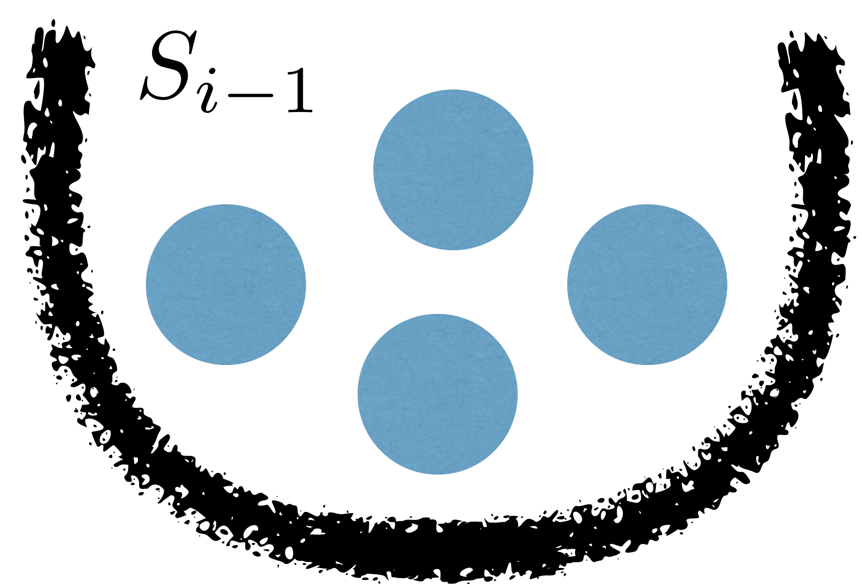
**Data Stream**

$u_i$ ←  ← $u_{i+1}$ $u_{i+2}$ $u_{i+3}$ $u_{i+4}$ $u_{i+5}$

Keep with probability $q = \frac{1}{p+\sqrt{p(p+1)}+1}$

$U_i \leftarrow \textsc{Exchange-Candidate}(S_{i-1}, u_i)$

**if** $f(u_i \mid S_{i-1}) \geq (1+c) \cdot f(U_i : S_{i-1})$
**then** Let $S_i \leftarrow S_{i-1} \setminus U_i + u_i$.



# Constrained Submodular Maximization

**Theorem 1: Non-monotone Submodular Maximization**

- The **Sample-Streaming** algorithm provides a solution for the problem of maximizing a **non-negative submodular function** $f$ subject to a $p$-matchoid constraint with a $(2p + 2\sqrt{p(p+1)} + 1)$-approximation guarantee
- The space complexity of this algorithm is $O(k)$
- The algorithm uses, in expectation, $O(km/p)$ value and independence oracle queries per each arriving element.

**Theorem 2: Monotone Submodular Maximization**

- The **Sample-Streaming** algorithm provides a solution for the problem of maximizing a **non-negative monotone submodular function** $f$ subject to a $p$-matchoid constraint with a $4p$-approximation guarantee
- The space complexity of this algorithm is $O(k)$
- The algorithm uses, in expectation, $O(km/p)$ value and independence oracle queries per each arriving element.

# Constrained Submodular Maximization

**Theorem 1: Non-monotone Submodular Maximization**

- The **Sample-Streaming** algorithm provides a solution for the problem of maximizing a **non-negative submodular function** $f$ subject to a $p$-matchoid constraint with a $(2p + 2\sqrt{p(p+1)} + 1)$-approximation guarantee
- The space complexity of this algorithm is $O(k)$
- The algorithm uses, in expectation, $O(km/p)$ value and independence oracle queries per each arriving element.

**Theorem 2: Monotone Submodular Maximization**

- The **Sample-Streaming** algorithm provides a solution for the problem of maximizing a **non-negative monotone submodular function** $f$ subject to a $p$-matchoid constraint with a $4p$-approximation guarantee
- The space complexity of this algorithm is $O(k)$
- The algorithm uses, in expectation, $O(km/p)$ value and independence oracle queries per each arriving element.

# Constrained Submodular Maximization

**Theorem 1: Non-monotone Submodular Maximization**

- The **Sample-Streaming** algorithm provides a solution for the problem of maximizing a **non-negative submodular function** $f$ subject to a $p$-matchoid constraint with a $(2p + 2\sqrt{p(p+1)} + 1)$-approximation guarantee
- The space complexity of this algorithm is $O(k)$
- The algorithm uses, in expectation, $O(km/p)$ value and independence oracle queries per each arriving element.

**Theorem 2: Monotone Submodular Maximization**

- The **Sample-Streaming** algorithm provides a solution for the problem of maximizing a **non-negative monotone submodular function** $f$ subject to a $p$-matchoid constraint with a $4p$-approximation guarantee
- The space complexity of this algorithm is $O(k)$
- The algorithm uses, in expectation, $O(km/p)$ value and independence oracle queries per each arriving element.

# Conclusion

| Algorithm | Function | Approx. Ratio | Memory | #Queries |
|---|---|---|---|---|
| Chekuri et al., 2015 | Monotone | $4p$ | $O(k)$ | $O(nkm)$ |
| Chekuri et al., 2015 (R) | Non-monotone | $\frac{5p+2+1/p}{1-\varepsilon}$ | $O(\frac{nk}{\varepsilon^2}\log\frac{k}{\varepsilon})$ | $O(\frac{nk^2m}{\varepsilon^2}\log\frac{k}{\varepsilon})$ |
| Chekuri et al., 2015 | Non-monotone | $\frac{9p+O(\sqrt{p})}{1-\varepsilon}$ | $O(\frac{k}{\varepsilon}\log\frac{k}{\varepsilon})$ | $O(\frac{nkm}{\varepsilon}\log\frac{k}{\varepsilon})$ |
| Local-Search | Non-monotone | $4p+4\sqrt{p}+1$ | $O(k\sqrt{p})$ | $O(n\sqrt{p}km)$ |
| Sample-Streaming (R) | Monotone | $4p$ | $O(k)$ | $O(nkm/p)$ |
| Sample-Streaming (R) | Non-monotone | $4p+2-o(1)$ | $O(k)$ | $O(nkm/p)$ |

- Our algorithm provides the best of three worlds:
  - the **tightest approximation guarantees** in various settings
  - **minimum memory** requirement
  - **fewest queries** per element

**Poster:** Today (Thu Dec 6th) 10:45 AM-12:45 PM @ Room 210 & 230 AB **#75**