Submodular Optimization: From Discrete to Continuous and Back

Hamed Hassani

Amin Karbasi

Penn

Yale

Slides + references: http://iid.yale.edu/icml/icml-20.md/

This Talk

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

Perfect Oracle

Discrete Methods

- Centralized
- Streaming
- Distributed

Imperfect Oracle

Continuous Methods

This Talk

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

Perfect Oracle

Discrete Methods

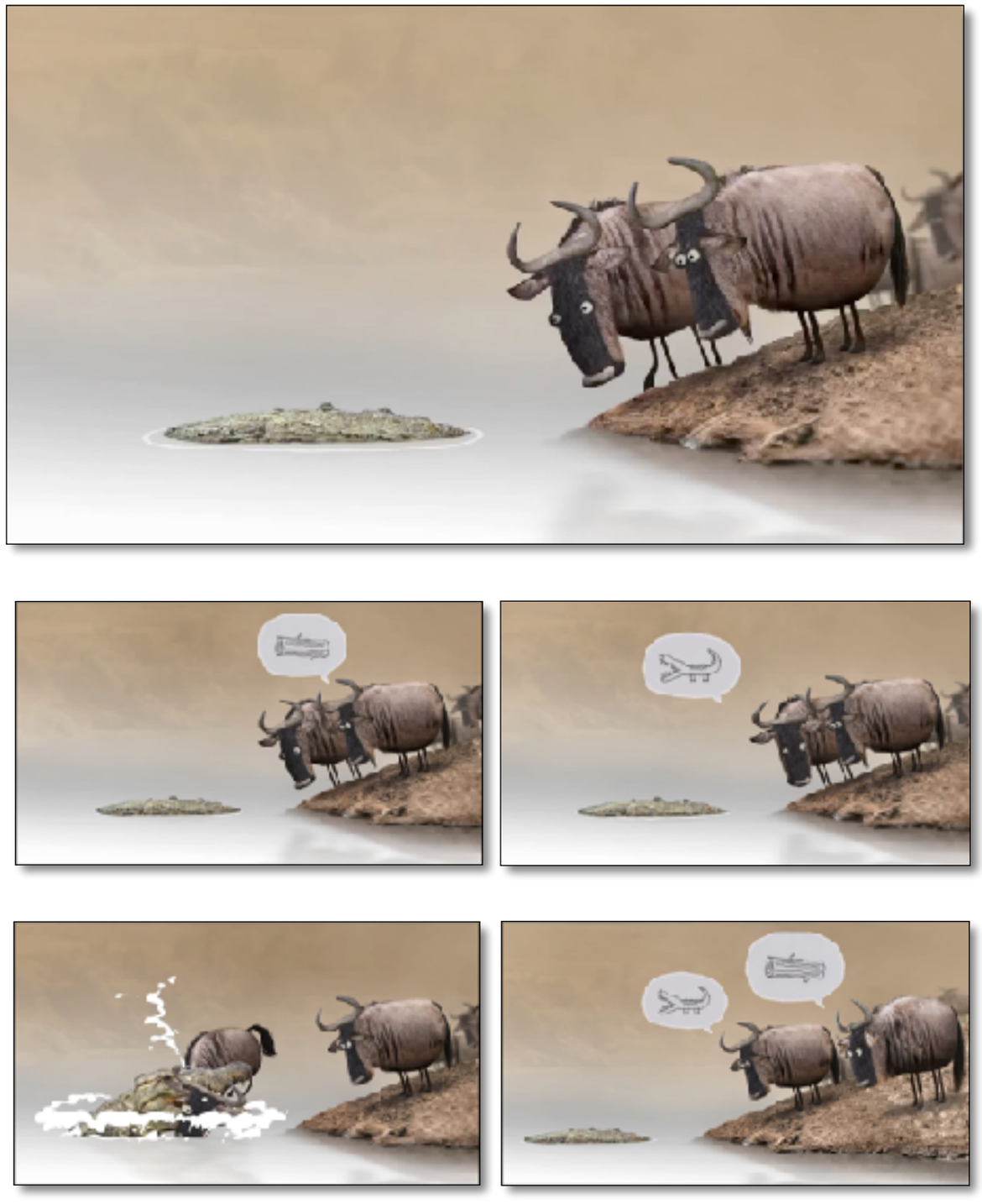
- **Centralized**
- **Streaming**
- **Distributed**

Imperfect Oracle

Continuous Methods

Data Stream

Data Stream



Problem: Greedy can be impractical

- Greedy approaches require random access to the complete data set; but for truly large-scale problems, where data is residing on disk, or arriving over time at a fast pace they are often impractical.

Problem: Greedy can be impractical

- Greedy approaches require random access to the complete data set; but for truly large-scale problems, where data is residing on disk, or arriving over time at a fast pace they are often impractical.

Is it possible to summarize a massive data set "on the fly": at any point of time we have access only to a small fraction of data?

$$S^* = \arg\max f(S)$$
$$S \subseteq V, S \in \mathcal{I}$$

Setting

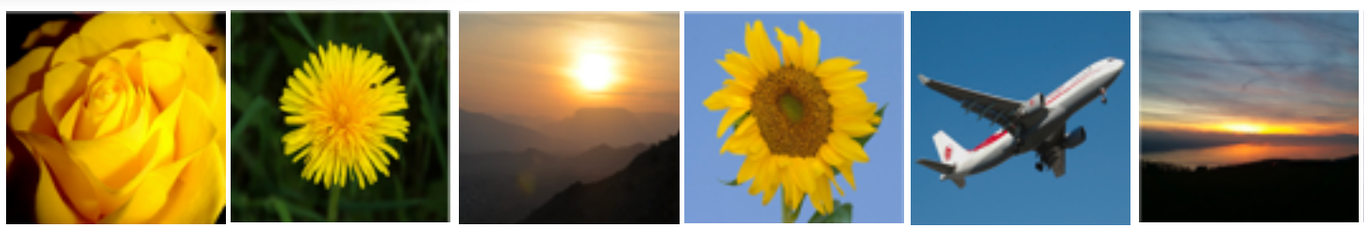
- Streaming

Stream Greedy

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Data Stream e



$$f(S \cup \{e\} \setminus \{x\}) - f(S) > 0$$

$$f\left(\left\{ \quad x_1 \quad x_2 \quad x_3 \quad \right\}\right) \; S$$

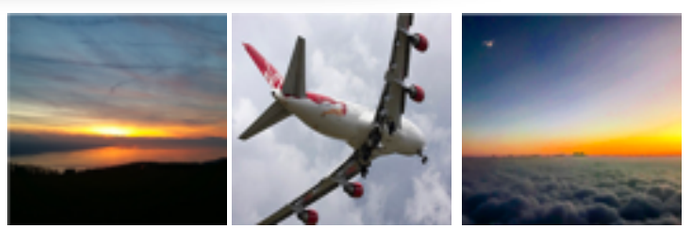
"Budgeted Nonparametric Learning from Data Stream", R. Gomes, A. Krause, 2010

Stream Greedy

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Data Stream e



$$f(S \cup \{e\} \setminus \{x\}) - f(S) > 0$$

$$f\left(\left\{ \quad \quad \quad \right\}\right) S$$

x_1 x_2 x_3

"Budgeted Nonparametric Learning from Data Stream", R. Gomes, A. Krause, 2010

Stream Greedy

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Data Stream e

$$f(S \cup \{e\} \setminus \{x\}) - f(S) > 0$$

$$f\left(\left\{ \quad \right\}\right) \; S$$

x_1 x_2 x_3

- Efficient in practice but in the worst case $f(S_{\text{Stream-Greedy}}) \leq \frac{1}{k}\text{OPT}$

"Budgeted Nonparametric Learning from Data Stream", R. Gomes, A. Krause, 2010

Bad Example

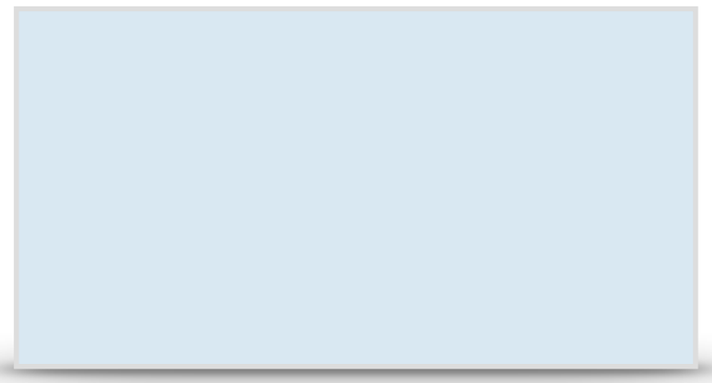
- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Stream Good

$$f(S) = \sum_{x \in \cup_{v \in S} v} w(x)$$

$$w(1) = \cdots = w(k) = 1$$
$$w(k+1) = \cdots = w(2k) = 1 + \epsilon$$
$$\cdots$$
$$w(k^2+1) = \cdots = w(k^2+k) = 1 + k\epsilon$$
$$\text{and } \epsilon \ll 1$$



Solution

Bad Example

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Stream Good

$$f(S) = \sum_{x \in \cup_{v \in S} v} w(x)$$

$$w(1) = \cdots = w(k) = 1$$
$$w(k+1) = \cdots = w(2k) = 1 + \epsilon$$
$$\cdots$$
$$w(k^2+1) = \cdots = w(k^2+k) = 1 + k\epsilon$$
$$\text{and } \epsilon \ll 1$$

{1} {2} {*k*}

Solution

Bad Example

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

$$f(S) = \sum_{x \in \cup_{v \in S} v} w(x)$$

$$w(1) = \cdots = w(k) = 1$$
$$w(k+1) = \cdots = w(2k) = 1 + \epsilon$$
$$\cdots$$
$$w(k^2+1) = \cdots = w(k^2+k) = 1 + k\epsilon$$
$$\text{and } \epsilon \ll 1$$

Stream Good

{1,2,…,*k*}

{1} {2} {*k*}

Solution

Bad Example

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

$$f(S) = \sum_{x \in \cup_{v \in S} v} w(x)$$

$$w(1) = \cdots = w(k) = 1$$
$$w(k+1) = \cdots = w(2k) = 1 + \epsilon$$
$$\cdots$$
$$w(k^2+1) = \cdots = w(k^2+k) = 1 + k\epsilon$$
$$\text{and } \epsilon \ll 1$$

Stream Good

{1,2,…,*k*}

{1} {2} {*k*}

Solution

Bad Example

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Stream Good

{1,2,…,*k*}

$$f(S) = \sum_{x \in \cup_{v \in S} v} w(x)$$

$$w(1) = \cdots = w(k) = 1$$
$$w(k+1) = \cdots = w(2k) = 1 + \epsilon$$
$$\cdots$$
$$w(k^2+1) = \cdots = w(k^2+k) = 1 + k\epsilon$$
$$\text{and } \epsilon \ll 1$$

{*k*+1} {2} {*k*}

Solution

Bad Example

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Stream Good

{1,2,…,*k*}

$$f(S) = \sum_{x \in \cup_{v \in S} v} w(x)$$

$$w(1) = \cdots = w(k) = 1$$
$$w(k+1) = \cdots = w(2k) = 1 + \epsilon$$
$$\cdots$$
$$w(k^2+1) = \cdots = w(k^2+k) = 1 + k\epsilon$$
$$\text{and } \epsilon \ll 1$$

{*k*+1} {*k*+2} {*k*}

Solution

Bad Example

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Stream Good

{1,2,…,*k*}

$$f(S) = \sum_{x \in \cup_{v \in S} v} w(x)$$

$$w(1) = \cdots = w(k) = 1$$
$$w(k+1) = \cdots = w(2k) = 1 + \epsilon$$
$$\cdots$$
$$w(k^2+1) = \cdots = w(k^2+k) = 1 + k\epsilon$$
$$\text{and } \epsilon \ll 1$$

{*k*+1} {*k*+2} {2*k*}

Solution

Bad Example

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

$$f(S) = \sum_{x \in \cup_{v \in S} v} w(x)$$

$$w(1) = \cdots = w(k) = 1$$
$$w(k+1) = \cdots = w(2k) = 1 + \epsilon$$
$$\cdots$$
$$w(k^2+1) = \cdots = w(k^2+k) = 1 + k\epsilon$$
$$\text{and } \epsilon \ll 1$$

Stream	Good
{*k*+1,*k*+2,…,2*k*}	{1,2,…,*k*}

{*k*+1} {*k*+2} {2*k*}

Solution

Bad Example

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

$$f(S) = \sum_{x \in \cup_{v \in S} v} w(x)$$

$$w(1) = \cdots = w(k) = 1$$
$$w(k+1) = \cdots = w(2k) = 1 + \epsilon$$
$$\cdots$$
$$w(k^2+1) = \cdots = w(k^2+k) = 1 + k\epsilon$$
$$\text{and } \epsilon \ll 1$$

Stream Good

{1,2,…,*k*}

{*k*+1,*k*+2,…,2*k*}

{*k*+1} {*k*+2} {2*k*}

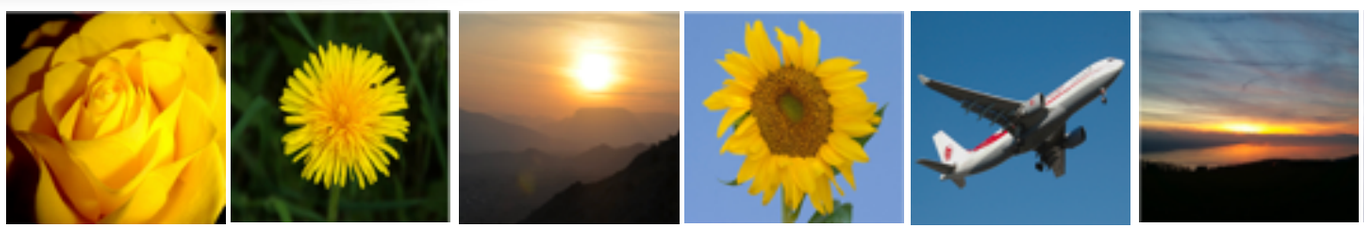
Solution

Thresholding

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Data Stream e



$$f(S \cup \{e\} \setminus \{x\}) - f(S) > 0$$

$$f\left(\left\{ \quad \right\}\right) S$$

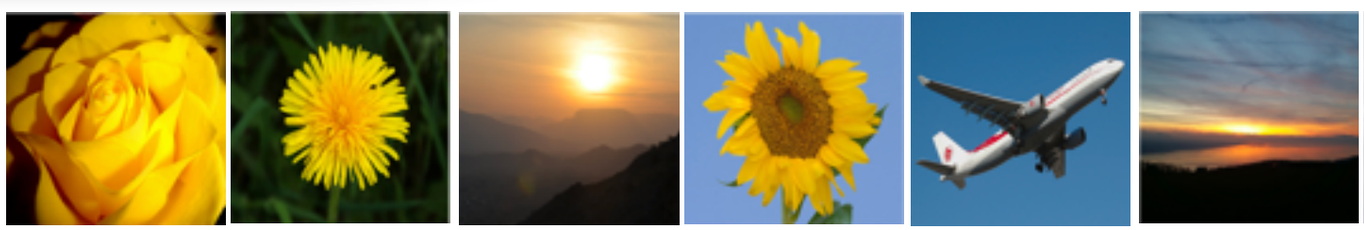
x_1 x_2 x_3

Thresholding

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Data Stream e



$$f(S \cup \{e\} \setminus \{x\}) - f(S) \geq \frac{f(S)}{k}$$

$f\left(\left\{\right.\right.$    $\left.\left.\right\}\right)$ S

x_1 x_2 x_3

Thresholding

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Data Stream e

$$f(S \cup \{e\} \setminus \{x\}) - f(S) \geq \frac{f(S)}{k}$$

$f\left(\left\{\right.\right.$    $\left.\left.\right\}\right) S$

x_1 x_2 x_3

Thresholding

- Consider the following problem where f is **monotone** submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Data Stream e

$$f(S \cup \{e\} \setminus \{x\}) - f(S) \geq \frac{f(S)}{k}$$

$$f\left(\left\{ \; , \; , \; \right\}\right) S$$

x_1 x_2 x_3

[Buchbinder, Feldman, Schwartz]

For monotone submodular functions, Threshold-Greedy, with memory O(k) achieves

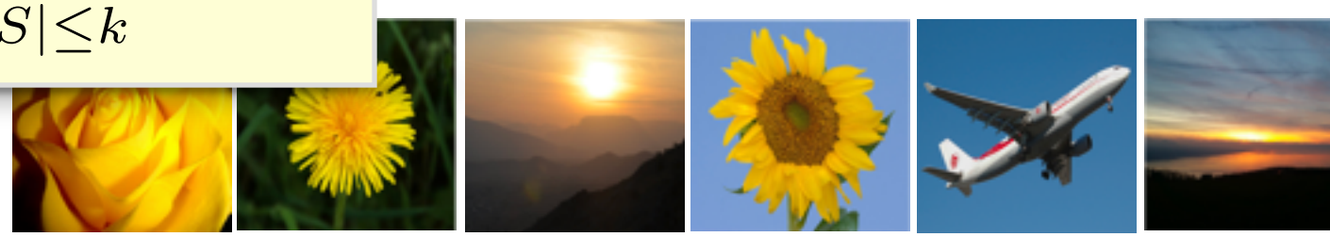
$$f(S_{\text{Threshold-Greedy}}) \geq \frac{1}{4}\text{OPT}$$

"Online Submodular Maximization with Preemption", 2015

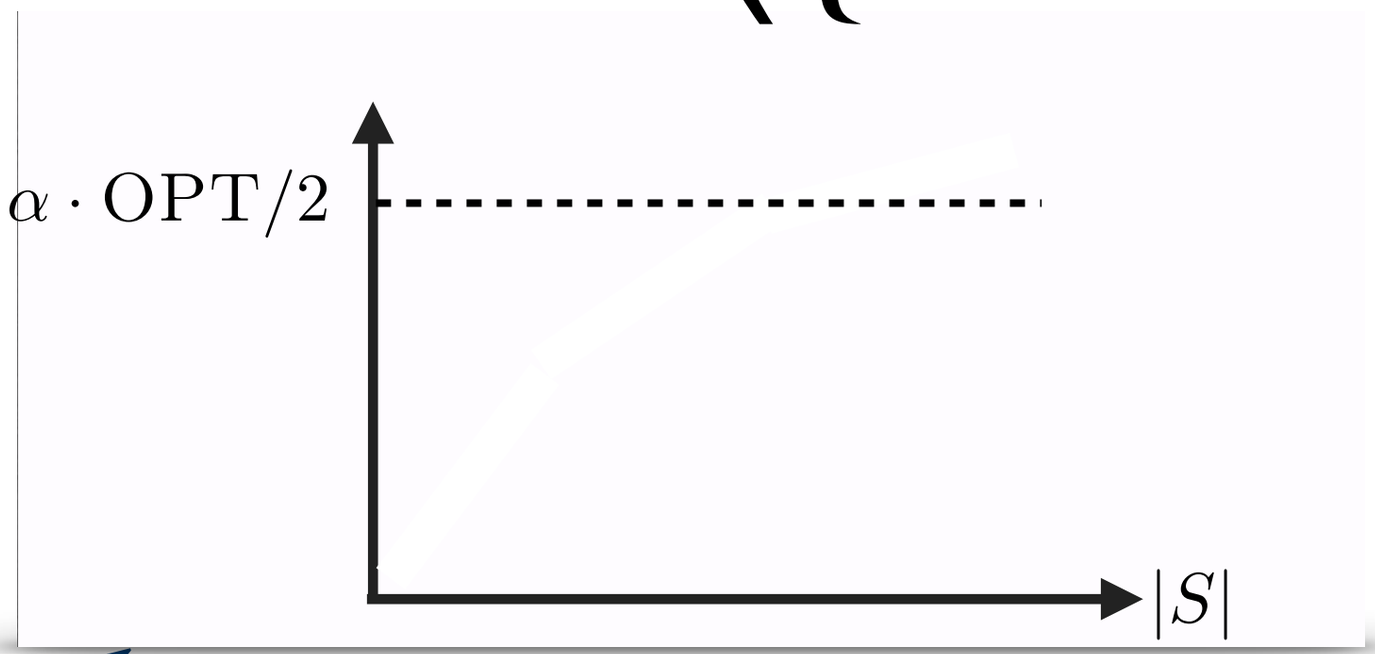
Thresholding + Guessing

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$



$$f(\{\quad\}) \longrightarrow \max$$



Thresholding + Guessing

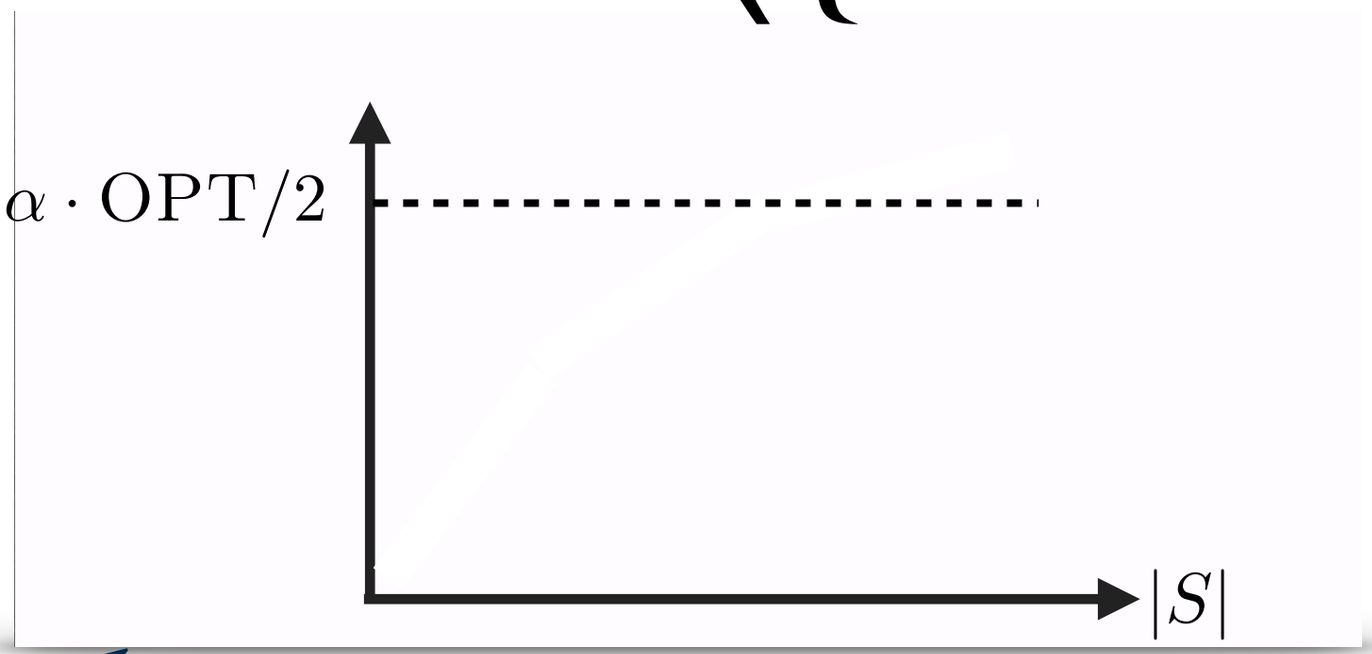
- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Data Stream

$$\mathrm{OPT} \geq \tau \geq \alpha \cdot \mathrm{OPT}$$
$$f(e_i \mid S) \geq \tau/(2k)$$

$f(\{\quad\}) \longrightarrow$ max



Thresholding + Guessing

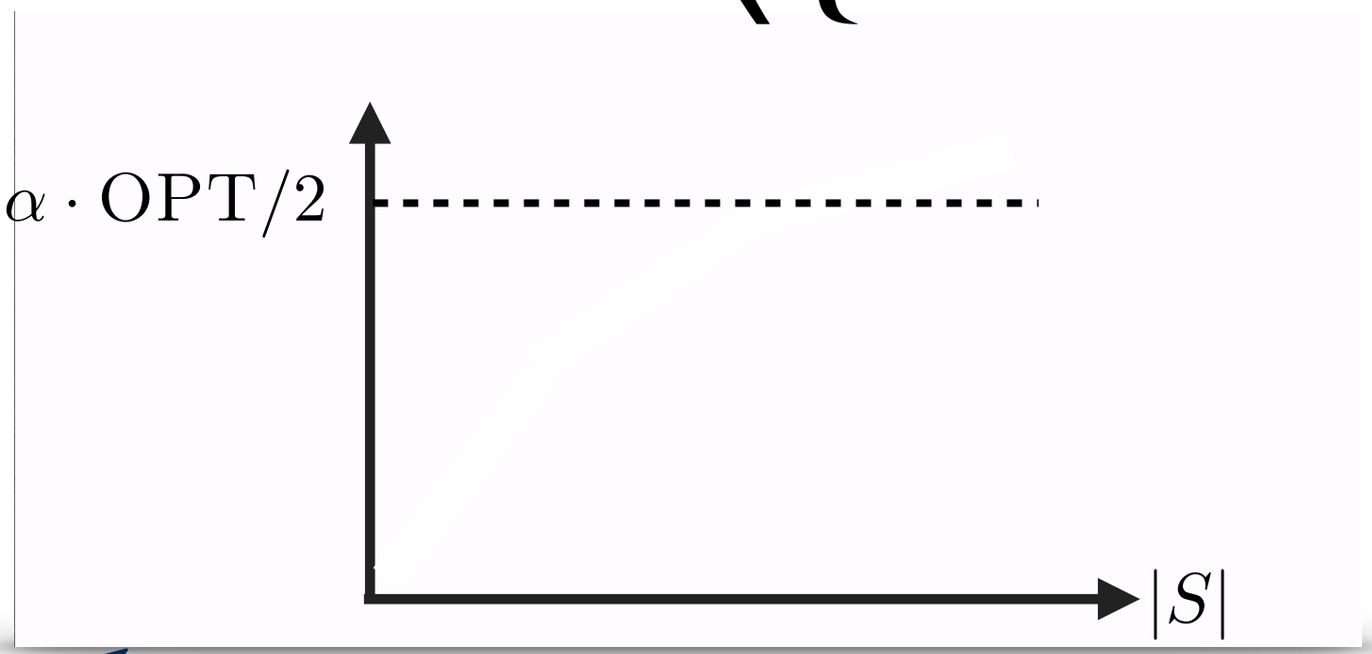
- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Data Stream

$$\mathrm{OPT} \geq \tau \geq \alpha \cdot \mathrm{OPT}$$
$$f(e_i \mid S) \geq \tau/(2k)$$

$$f(\{\quad\}) \longrightarrow \max$$



Thresholding + Guessing

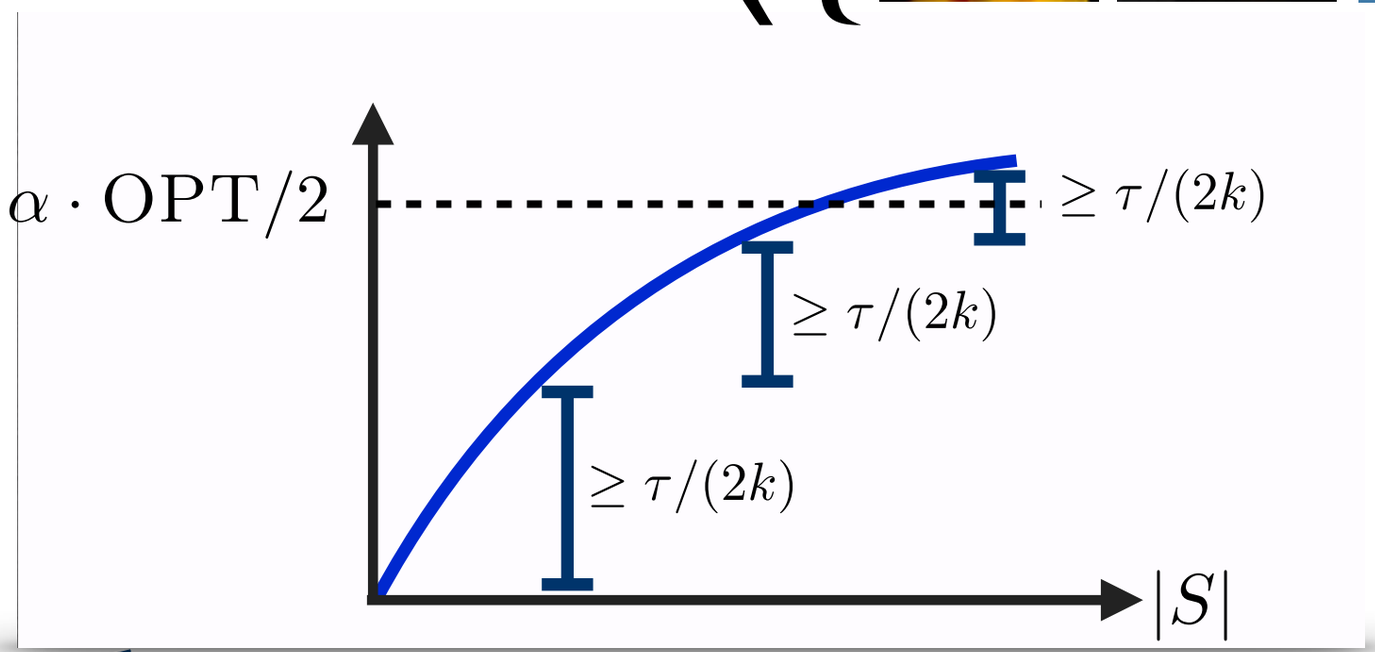
- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Data Stream

$$\mathrm{OPT} \geq \tau \geq \alpha \cdot \mathrm{OPT}$$
$$f(e_i \mid S) \geq \tau/(2k)$$

$$f\left(\left\{\right.\right.$$  $$\left.\left.\right\}\right) \longrightarrow \max$$



Properties:

- 1 pass
- $f(S) \geq \alpha \cdot \mathrm{OPT}/2$
- $O(k)$ memory
- $O(1)$ update time

Thresholding + Guessing

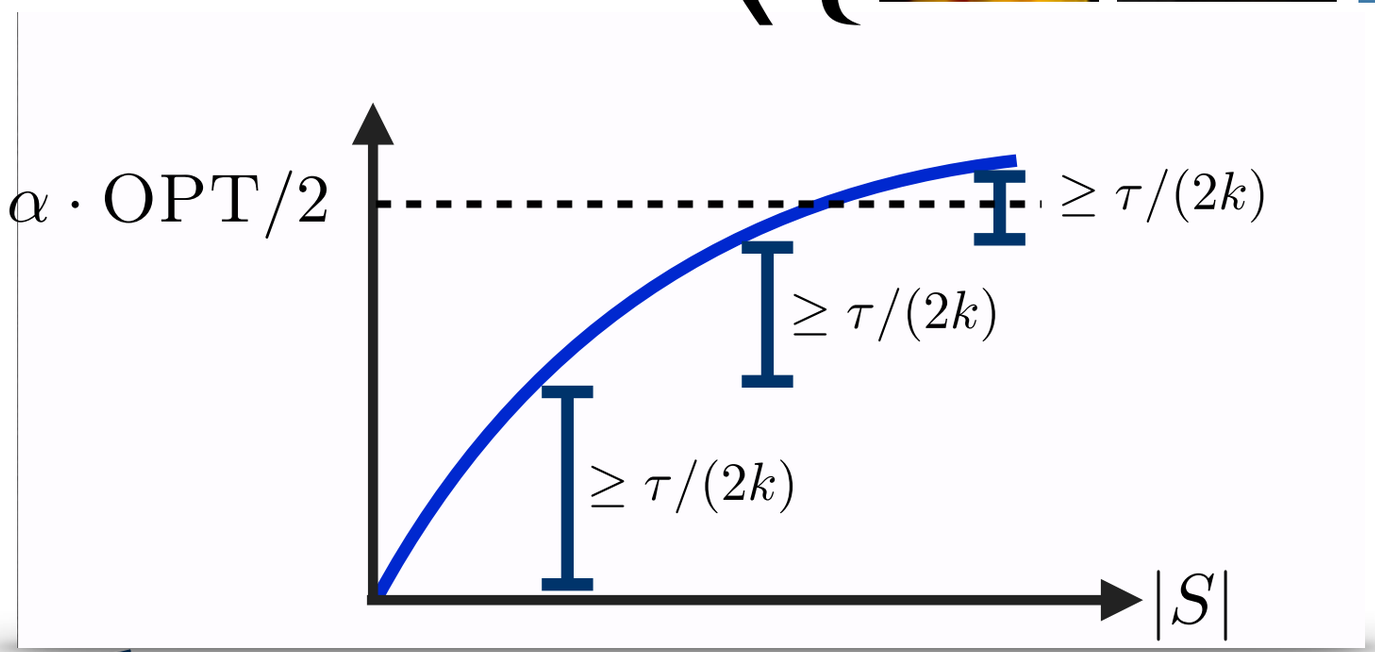
- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Data Stream

$$\mathrm{OPT} \geq \tau \geq \alpha \cdot \mathrm{OPT}$$
$$f(e_i \mid S) \geq \tau/(2k)$$

$$f\left(\left\{\right.\right.$$  $$\left.\left.\right\}\right) \longrightarrow \max$$



Properties:

- 1 pass
- $f(S) \geq \alpha \cdot \mathrm{OPT}/2$
- $O(k)$ memory
- $O(1)$ update time

Knowing max marginal is enough

Knowing max marginal is enough

- Can we find a good approximation OPT?

Knowing max marginal is enough

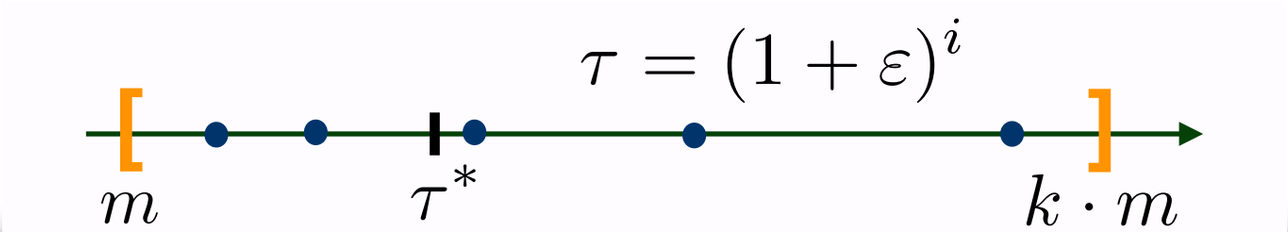
- Can we find a good approximation OPT?
  - Submodularity: $m \leq \mathrm{OPT} \leq k \cdot m, m = \max_{e \in V} f(\{e\})$

Knowing max marginal is enough

- Can we find a good approximation OPT?
  - Submodularity: $m \leq \mathrm{OPT} \leq k \cdot m, m = \max_{e \in V} f(\{e\})$

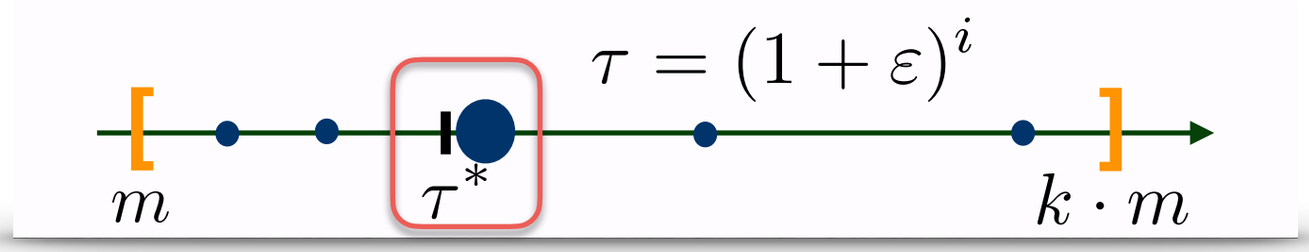
Knowing max marginal is enough

- Can we find a good approximation OPT?
  - Submodularity: $m \leq \mathrm{OPT} \leq k \cdot m, m = \max_{e \in V} f(\{e\})$
  - Discretizing $[m, k \cdot m]$



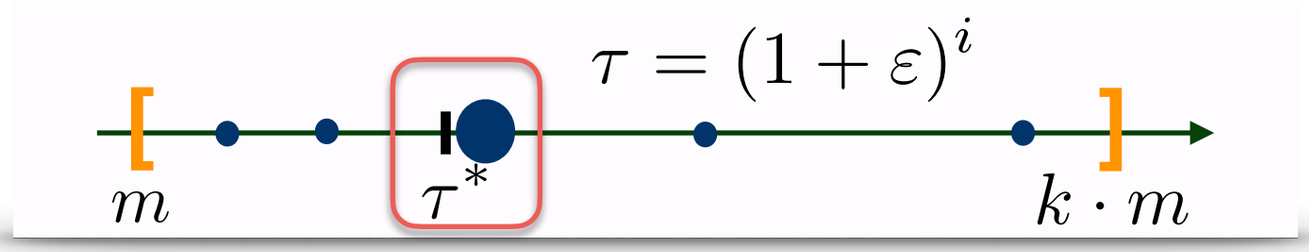
Knowing max marginal is enough

- Can we find a good approximation OPT?
  - Submodularity: $m \leq \mathrm{OPT} \leq k \cdot m, m = \max_{e \in V} f(\{e\})$
  - Discretizing $[m, k \cdot m]$
  - At least one τ should be a good estimate of OPT

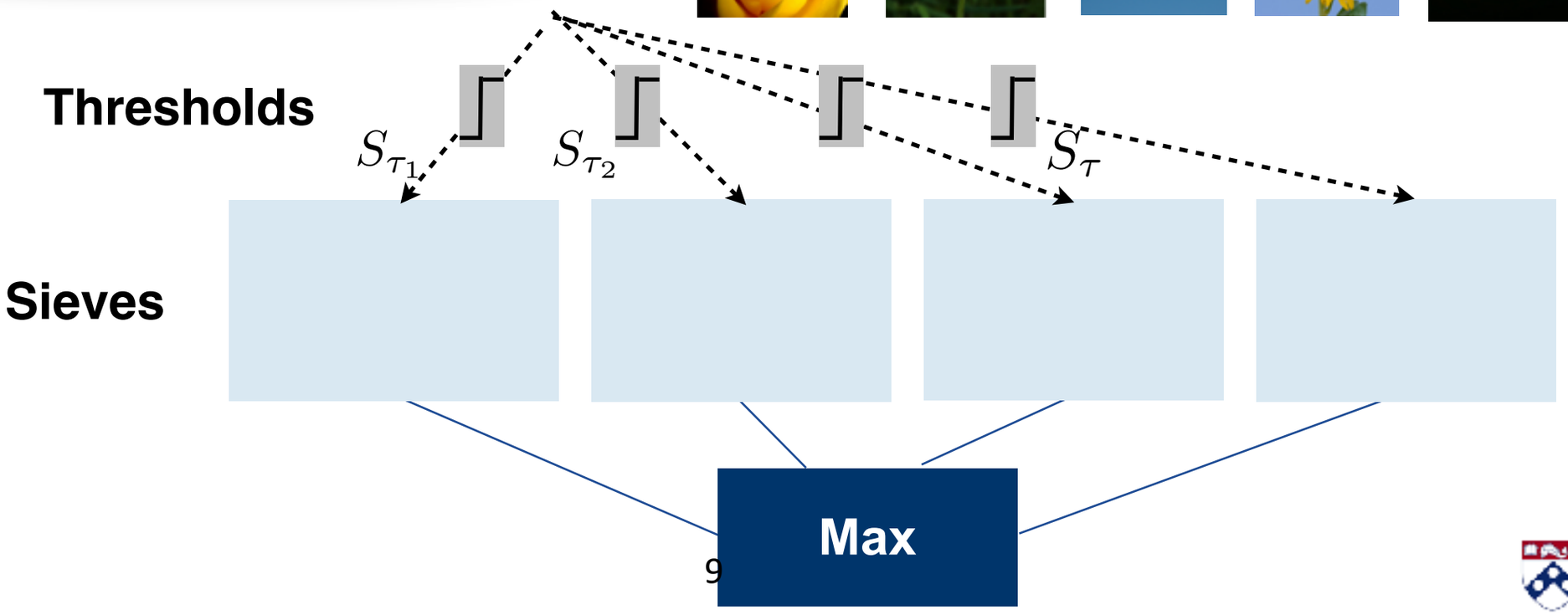


Knowing max marginal is enough

- Can we find a good approximation OPT?
  - Submodularity: $m \leq \mathrm{OPT} \leq k \cdot m, m = \max_{e \in V} f(\{e\})$
  - Discretizing $[m, k \cdot m]$
  - At least one τ should be a good estimate of OPT

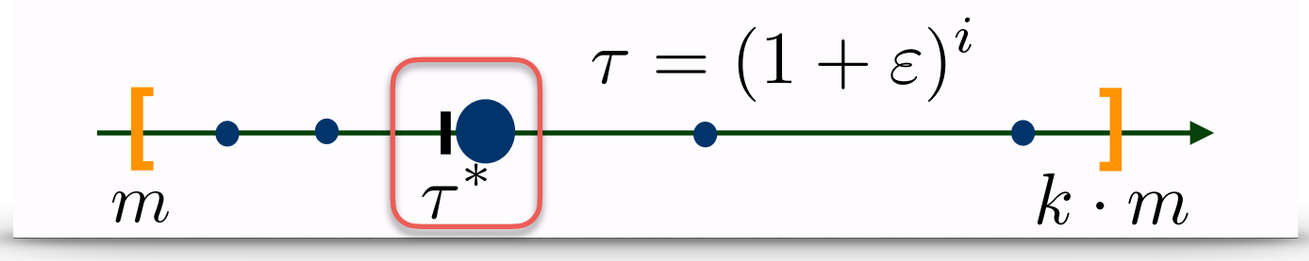


$f(e_i \mid S) \geq \tau/(2k)$

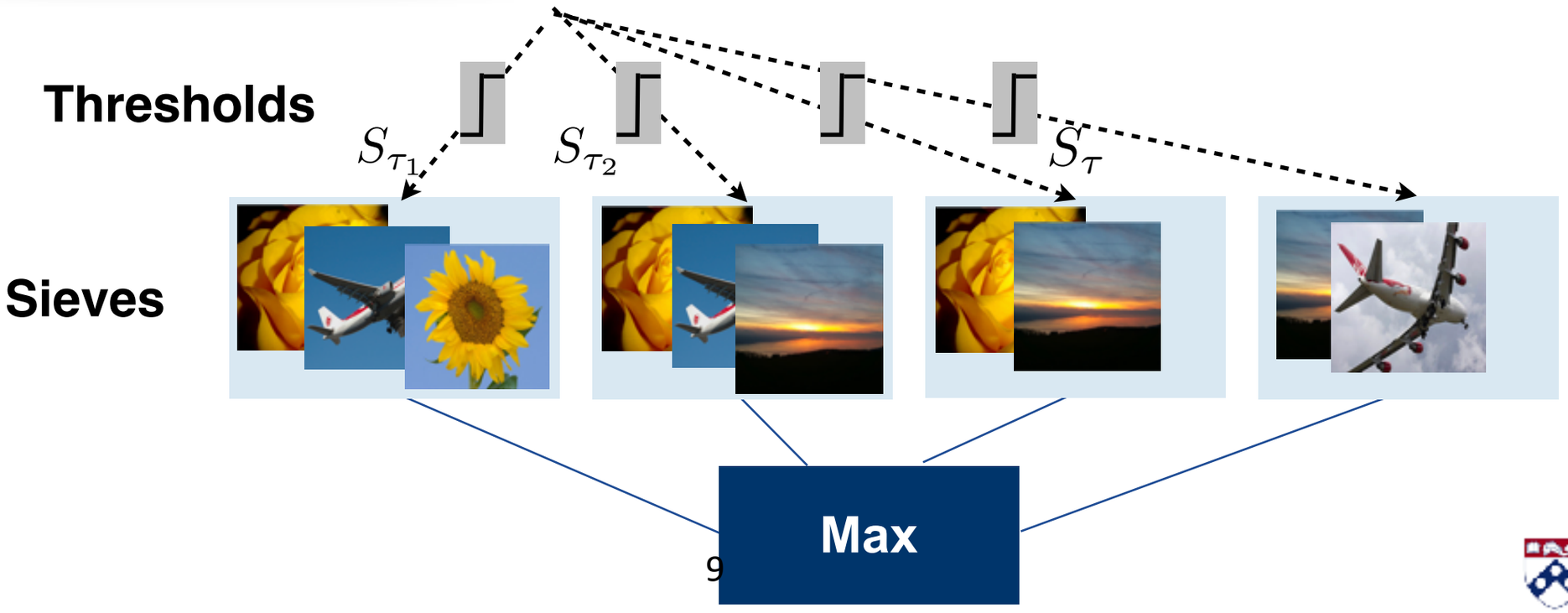


Knowing max marginal is enough

- Can we find a good approximation OPT?
  - Submodularity: $m \leq \mathrm{OPT} \leq k \cdot m, m = \max_{e \in V} f(\{e\})$
  - Discretizing $[m, k \cdot m]$
  - At least one τ should be a good estimate of OPT

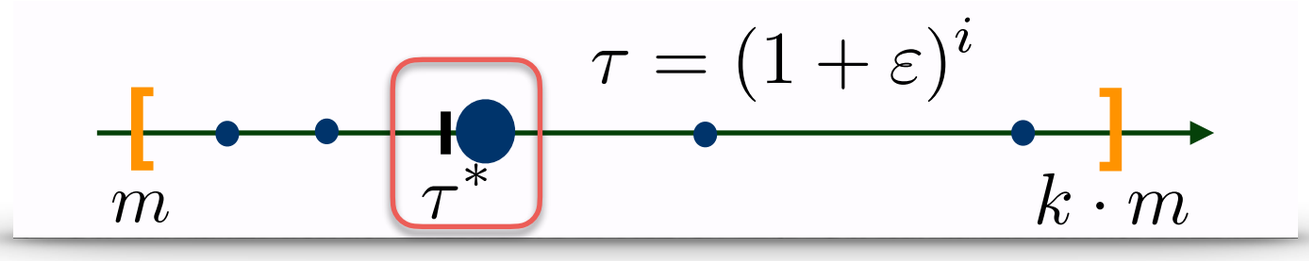


$f(e_i \mid S) \geq \tau/(2k)$



Knowing max marginal is enough

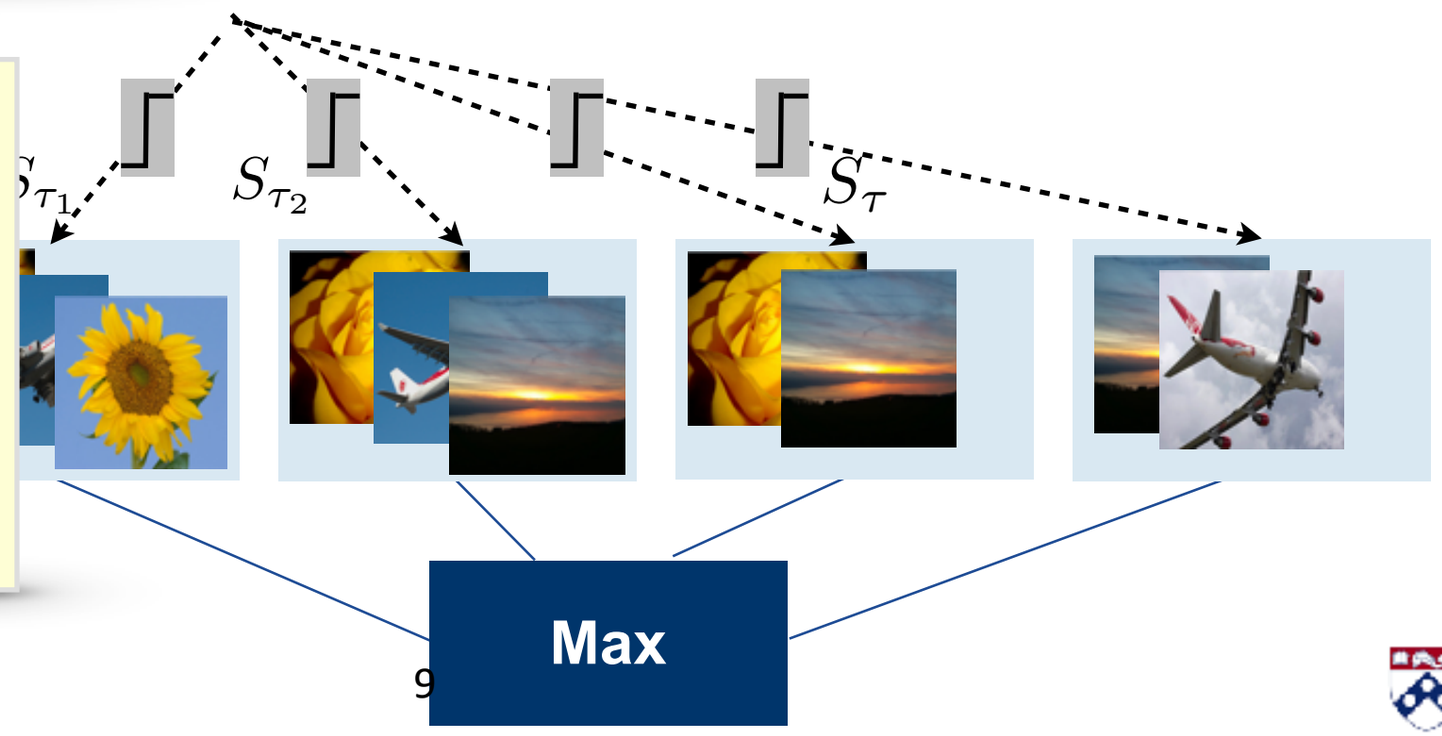
- Can we find a good approximation OPT?
  - Submodularity: $m \leq \mathrm{OPT} \leq k \cdot m, m = \max_{e \in V} f(\{e\})$
  - Discretizing $[m, k \cdot m]$
  - At least one τ should be a good estimate of OPT



$f(e_i \mid S) \geq \tau/(2k)$

Properties:

- 1 pass
- $f(S) \geq (1/2 - \varepsilon)\mathrm{OPT}$
- $O(k \log k/\varepsilon)$ memory
- $O(\log k/\varepsilon)$ update time



SIEVE-STREAMING

[Badanidiyuru, Mirzasoleiman, Karbasi, Krause]

For monotone submodular functions, SIEVE-STREAMING with a $O(k\log(k)/\varepsilon)$ memory achieves

$$f(S_{\text{Sieve-Streaming}}) \geq \left(\frac{1}{2} - \epsilon\right) \text{OPT}$$

"Streaming Submodular Maximization: Massive Data Summarization on the Fly", 2014

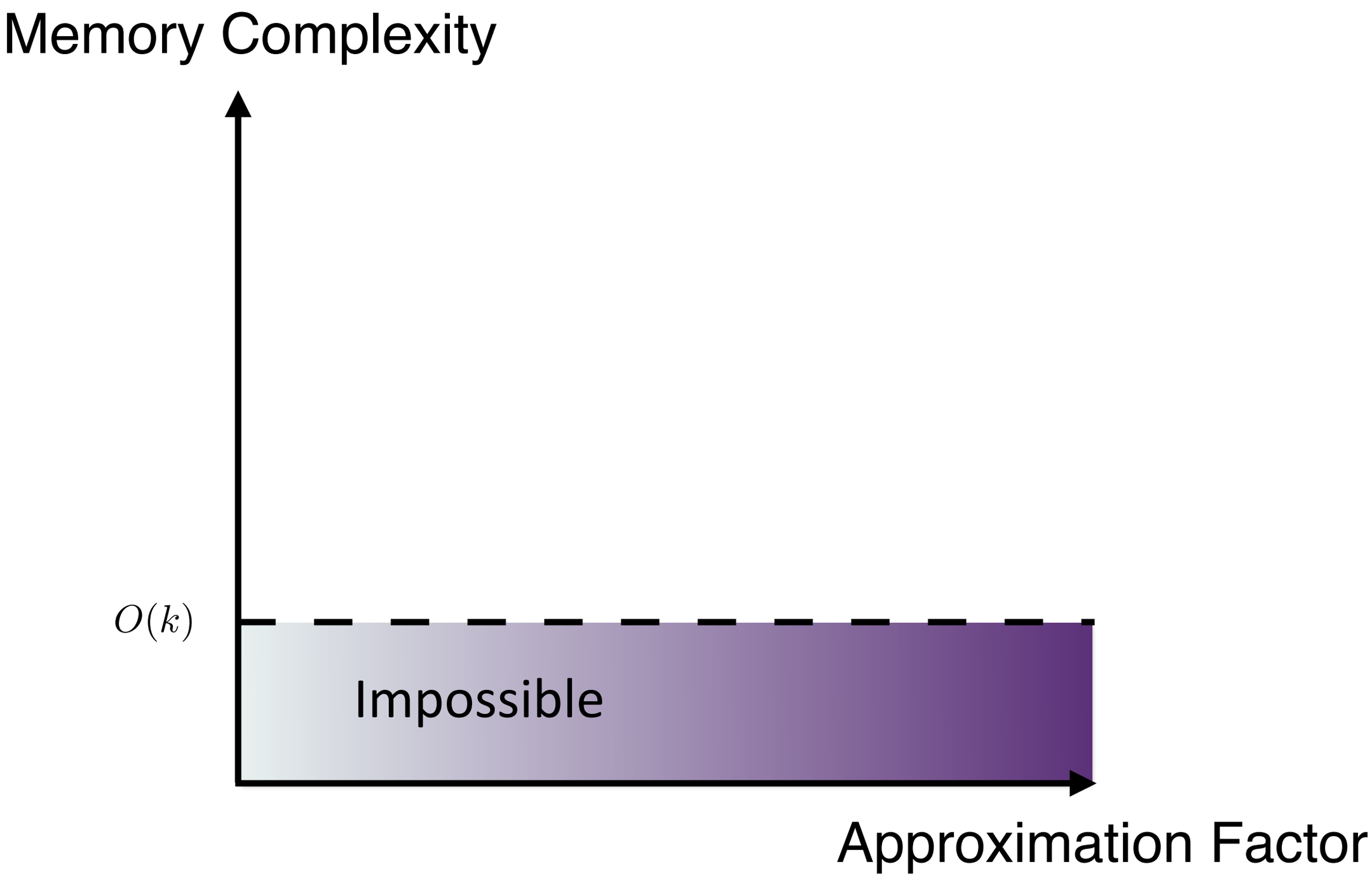
[Kazemi, Mitrovic, Zadimoghaddam, Lattanzi, Karbasi]

For monotone submodular functions, SIEVE-STREAMING++ with a $O(k)$ memory achieves

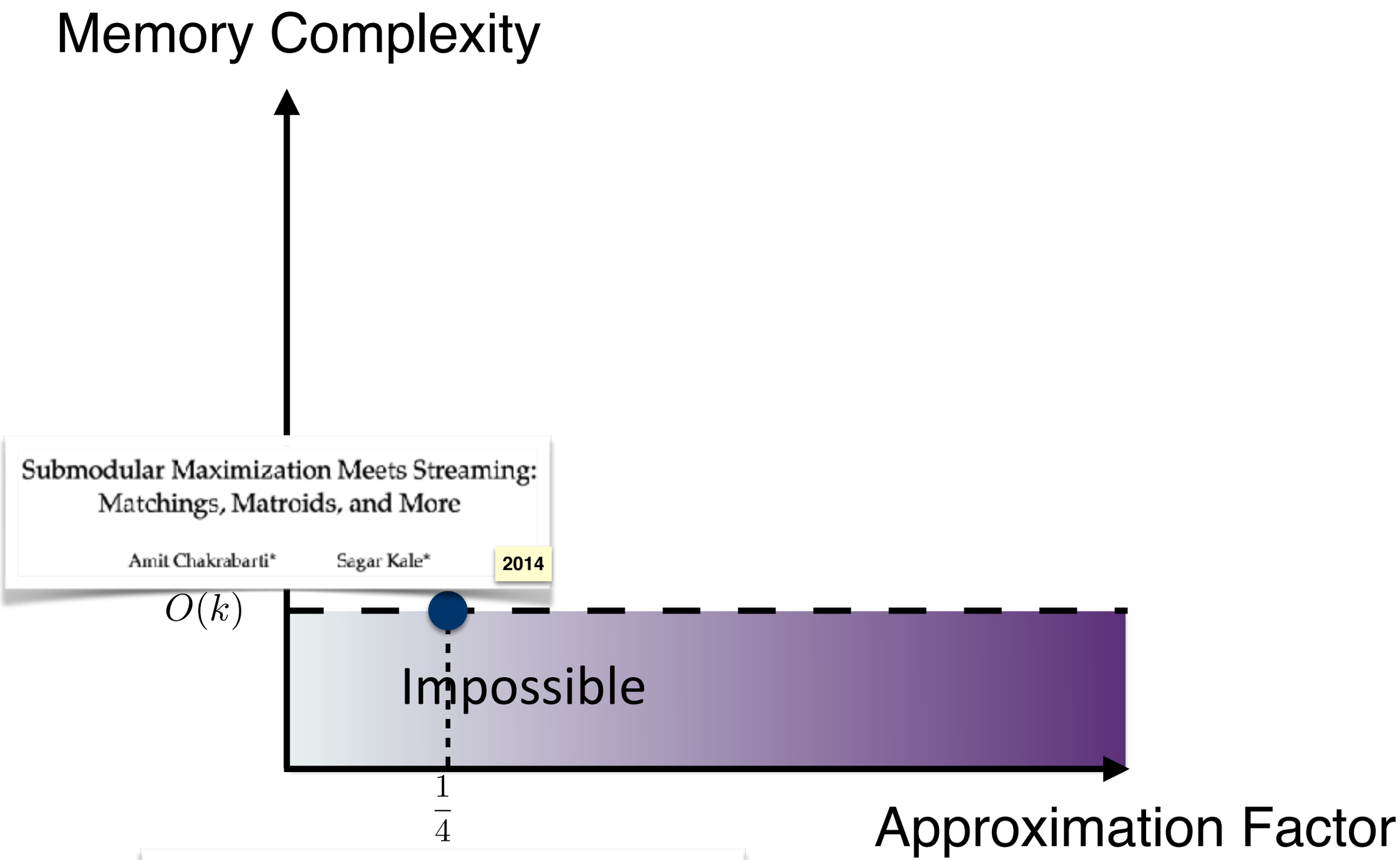
$$f(S_{\text{Sieve-Streaming++}}) \geq \left(\frac{1}{2} - \epsilon\right) \text{OPT}$$

"Submodular Streaming in All Its Glory: Tight Approximation, Minimum Memory and Low Adaptive Complexity", 2019

Streaming Algorithms: Cardinality Constraint



Streaming Algorithms: Cardinality Constraint



Yale

Penn

Streaming Algorithms: Cardinality Constraint

Memory Complexity

Streaming Submodular Maximization: Massive Data Summarization on the Fly

Ashwinkumar Badanidiyuru
Cornell Unversity
ashwinkumarbv@gmail.com

Baharan Mirzasoleiman
ETH Zurich
baharanm@inf.ethz.ch

Amin Karbasi
ETH Zurich
Yale University
amin.karbasi@gmail.com

Andreas Krause
ETH Zurich
krausea@ethz.ch

2014

$O(k \log k)$

Submodular Maximization Meets Streaming: Matchings, Matroids, and More

Amit Chakrabarti* Sagar Kale*

2014

$O(k)$

Impossible

$\frac{1}{4}$ $\frac{1}{2} - \epsilon$

Approximation Factor

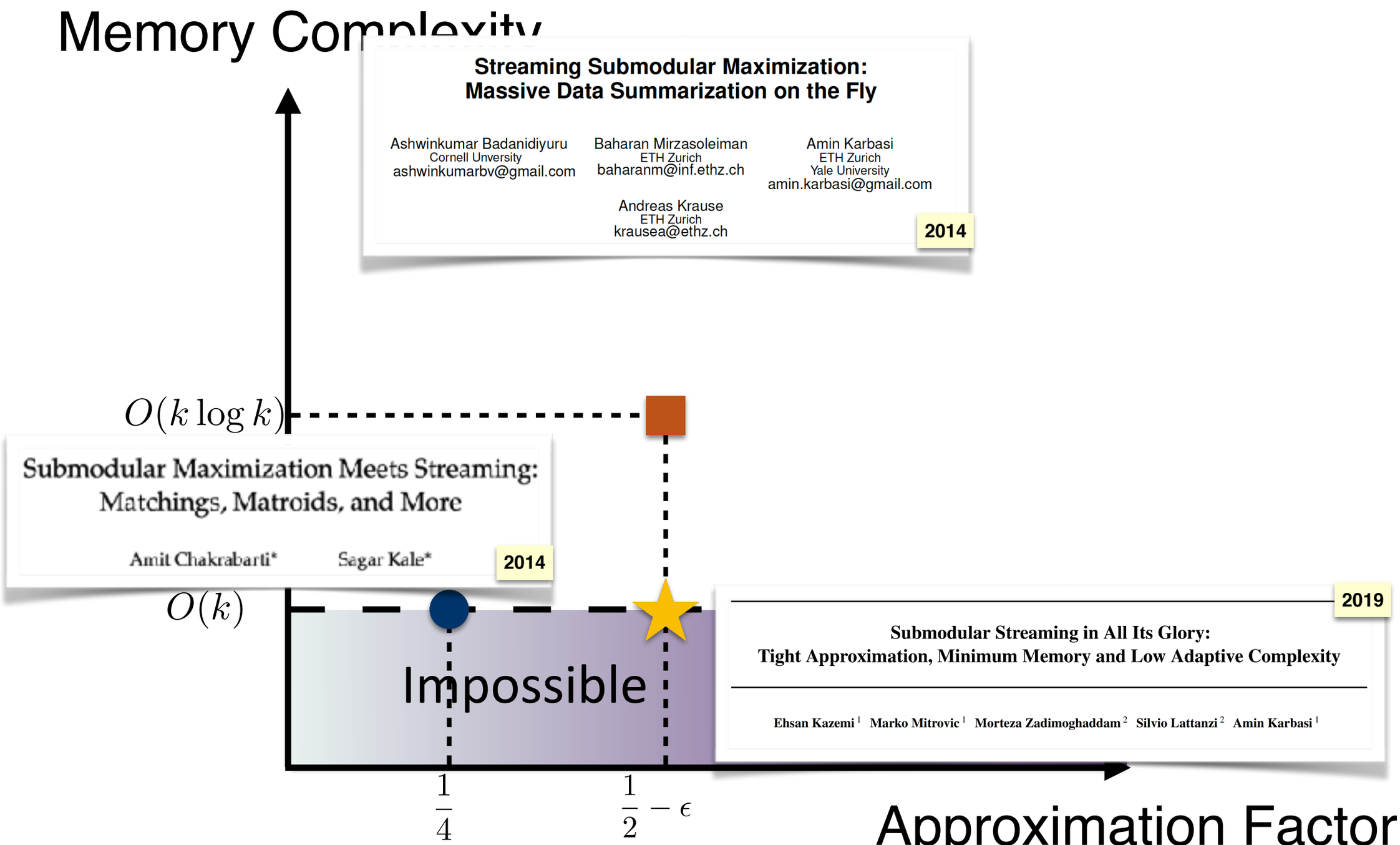
Online Submodular Maximization with Preemption

Niv Buchbinder* Moran Feldman† Roy Schwartz‡ 2015

Yale

Penn

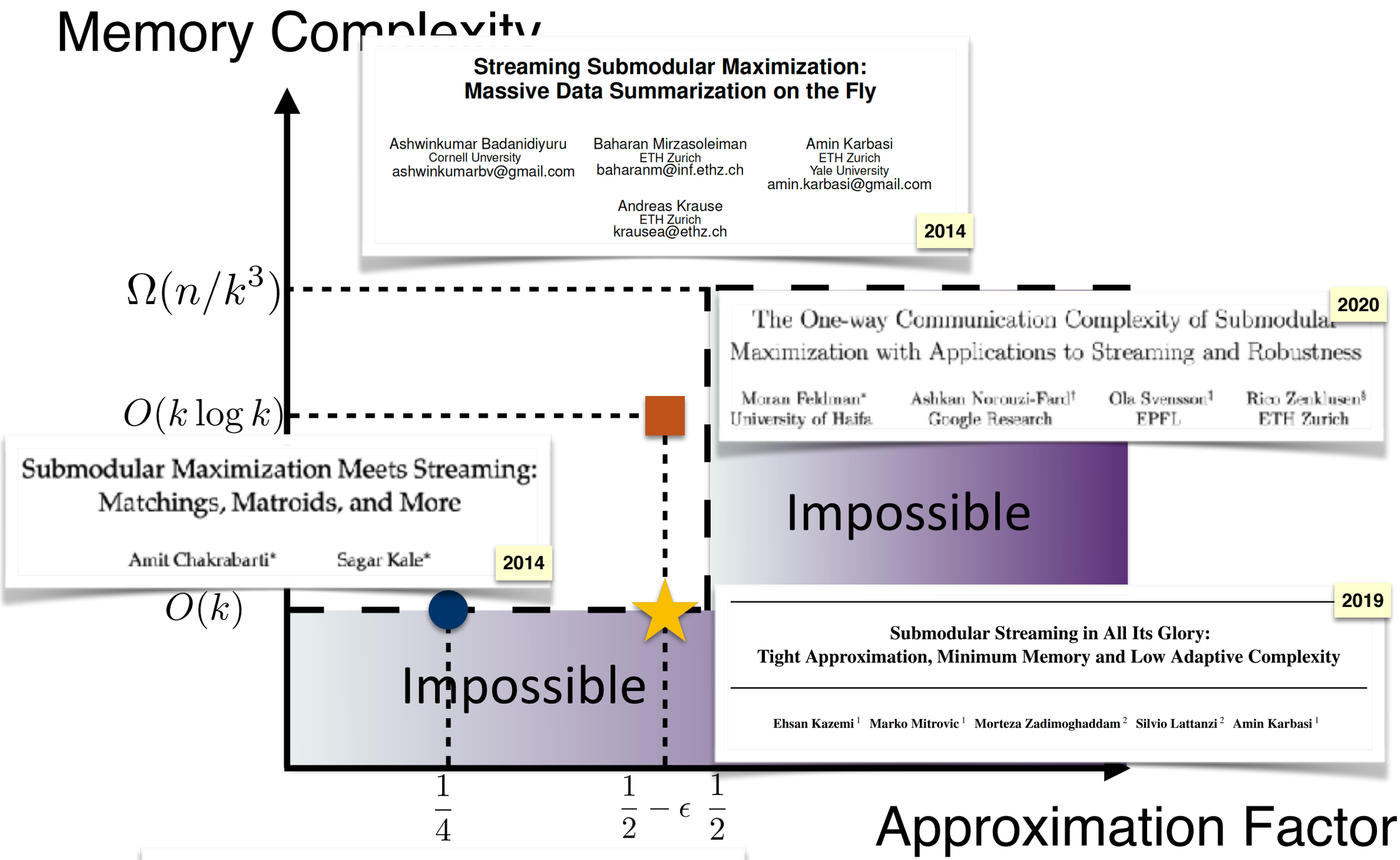
Streaming Algorithms: Cardinality Constraint



Yale

Penn

Streaming Algorithms: Cardinality Constraint

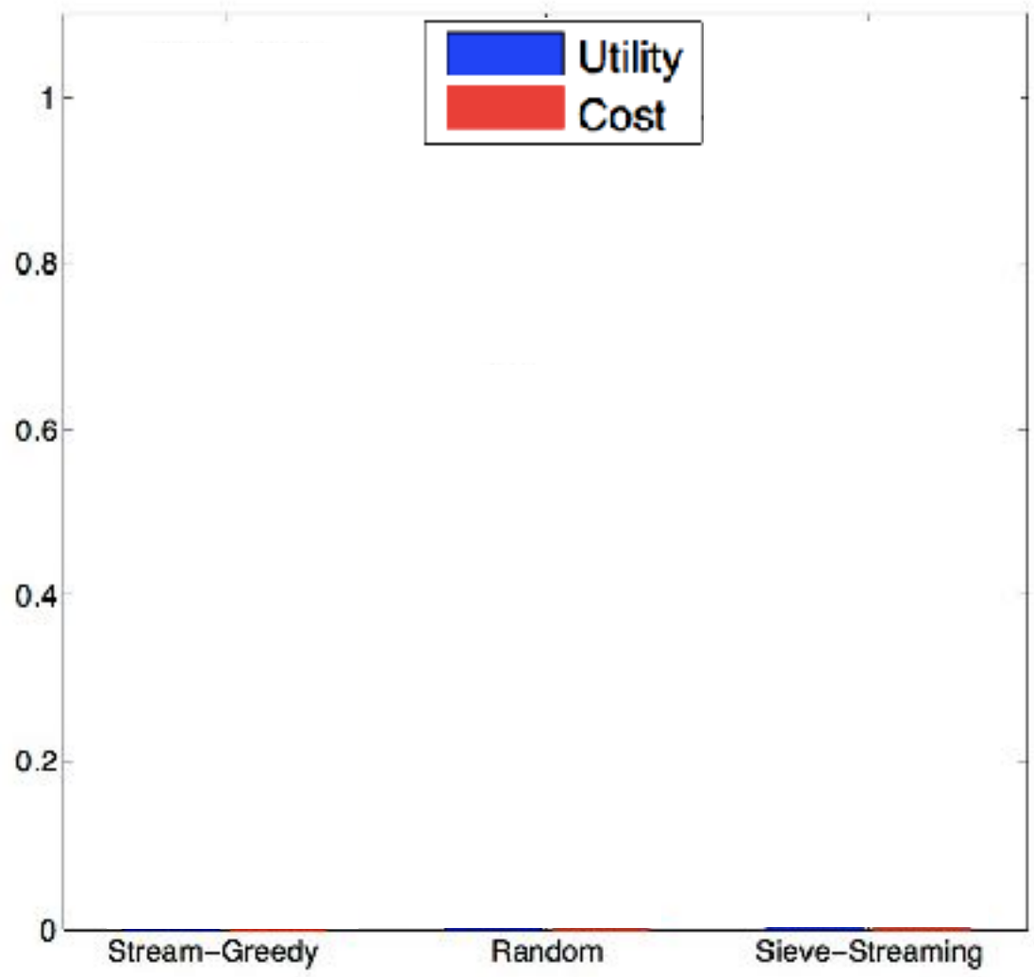


Yale

Penn

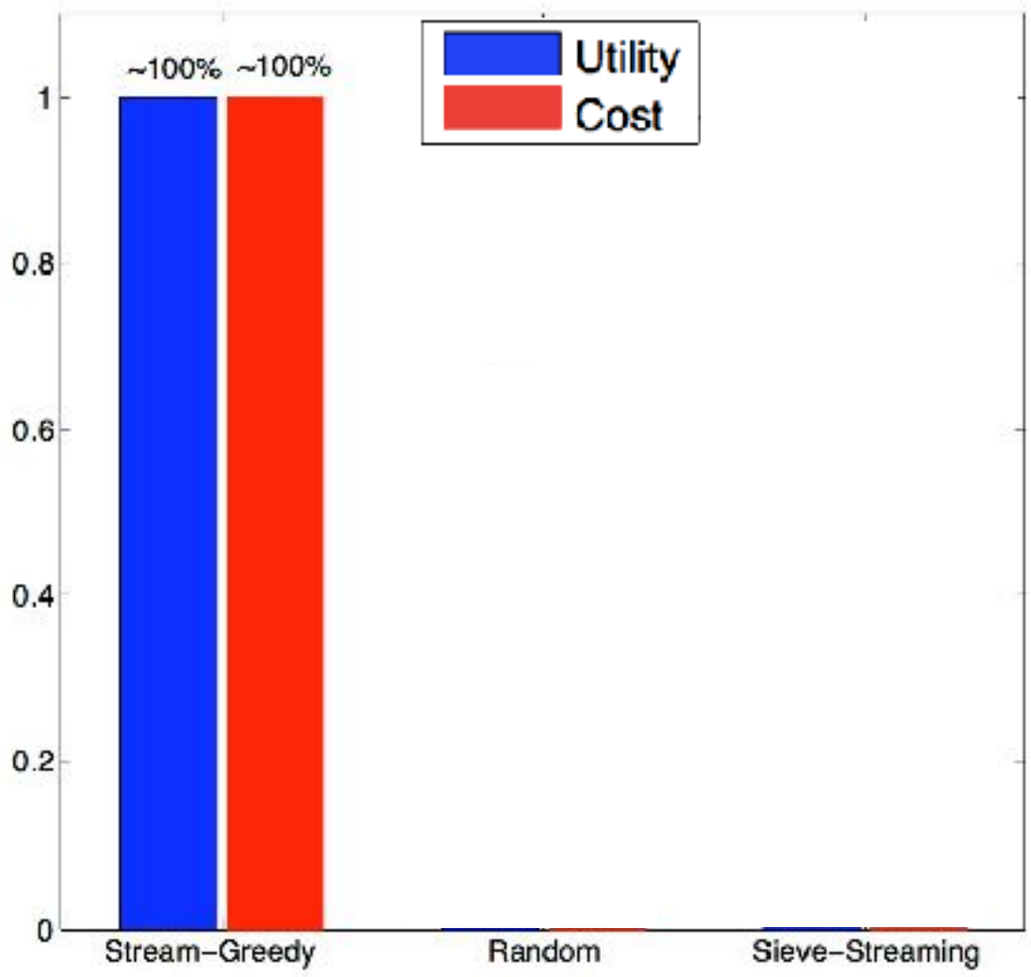
Nonparametric Regression

- **Census** dataset consists of 2,458,285 data points with 68 attributes



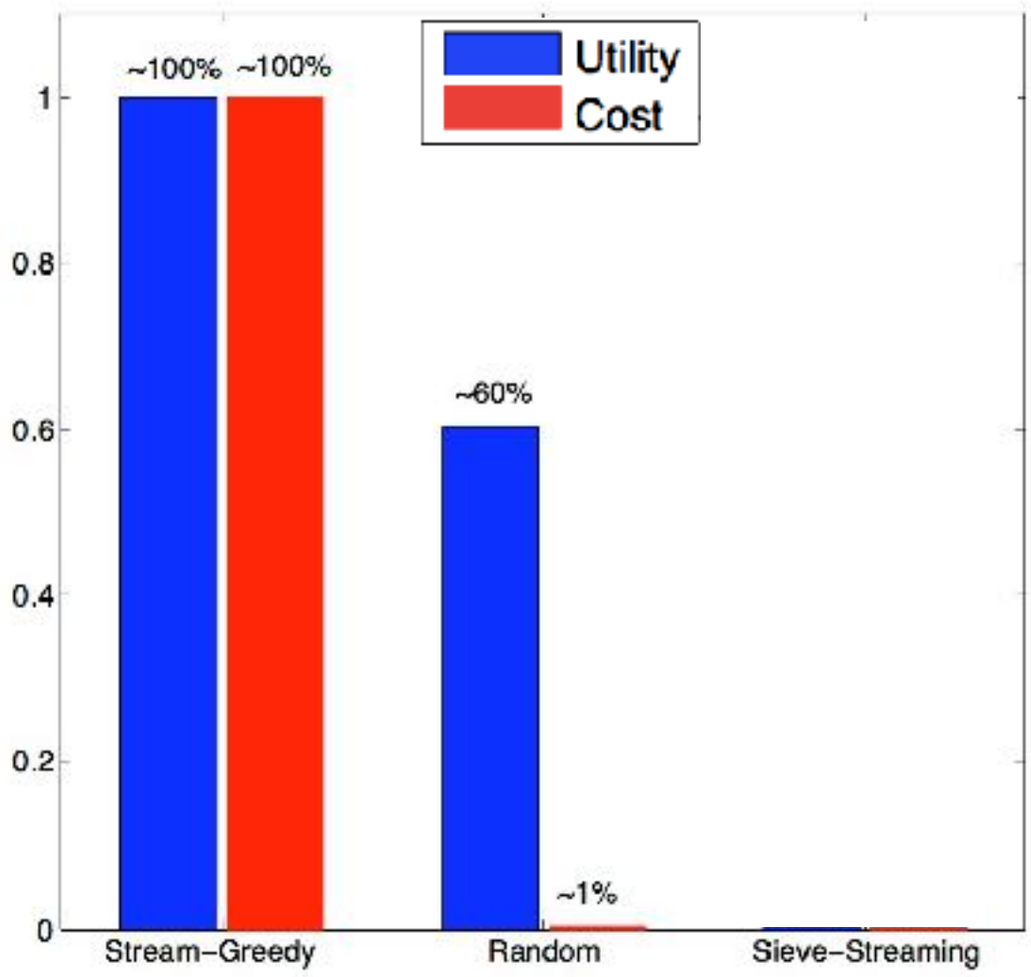
Nonparametric Regression

- **Census** dataset consists of 2,458,285 data points with 68 attributes



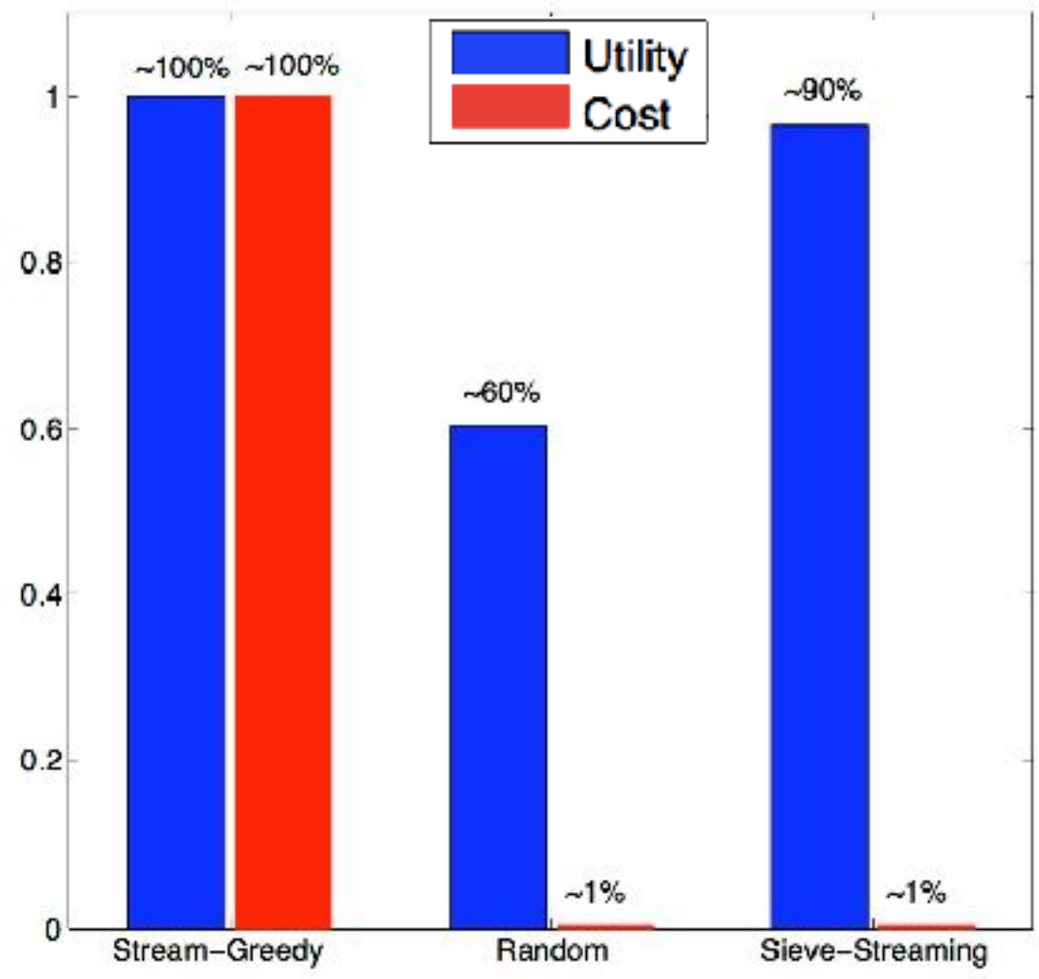
Nonparametric Regression

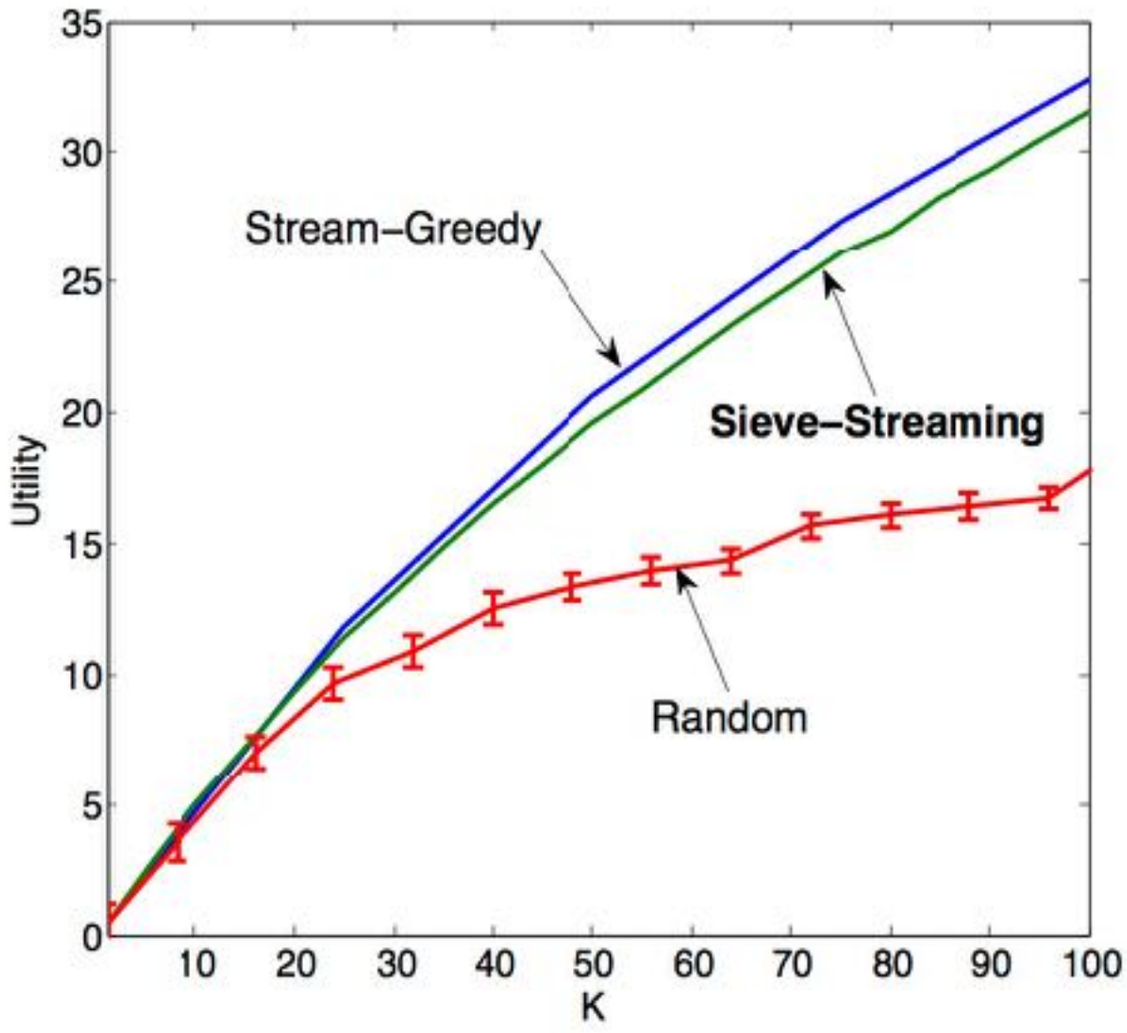
- **Census** dataset consists of 2,458,285 data points with 68 attributes



Nonparametric Regression

- **Census** dataset consists of 2,458,285 data points with 68 attributes





Open Question: Matroid

[Chakrabati, Kale]

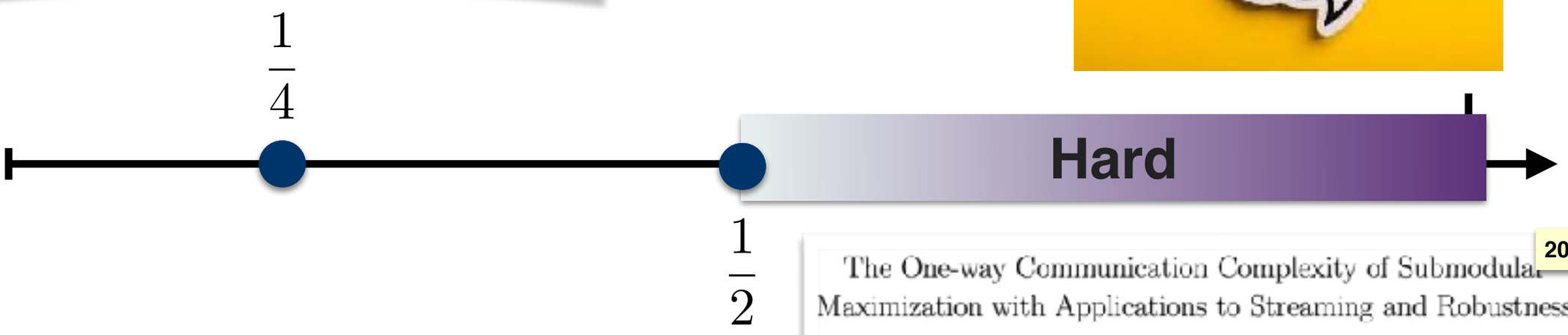
For monotone submodular functions, and subject to the intersection of p matroids, there is a one-pass streaming algorithm that achieves

$$f(S_{\text{MSIS}}) \geq \left(\frac{1}{4p}\right) \text{OPT}$$

Submodular Maximization Meets Streaming: Matchings, Matroids, and More

Amit Chakrabarti* Sagar Kale*

2014



The One-way Communication Complexity of Submodular Maximization with Applications to Streaming and Robustness

Moran Feldman* University of Haifa
Ashkan Norouzi-Fard† Google Research
Ola Svensson‡ EPFL
Rico Zenklusen§ ETH Zurich

2020

Open Question: Matroid

[Chakrabati, Kale]

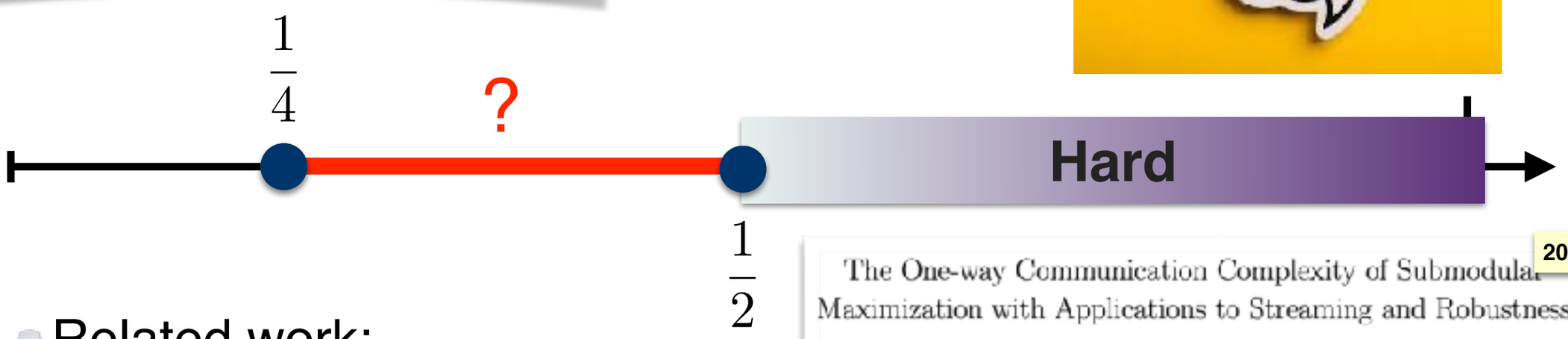
For monotone submodular functions, and subject to the intersection of p matroids, there is a one-pass streaming algorithm that achieves

$$f(S_{\mathrm{MSIS}}) \geq \left(\frac{1}{4p}\right) \mathrm{OPT}$$

Submodular Maximization Meets Streaming: Matchings, Matroids, and More

Amit Chakrabarti* Sagar Kale*

2014



- Related work:

The One-way Communication Complexity of Submodular Maximization with Applications to Streaming and Robustness

Moran Feldman* University of Haifa
Ashkan Norouzi-Fard† Google Research
Ola Svensson‡ EPFL
Rico Zenklusen§ ETH Zurich

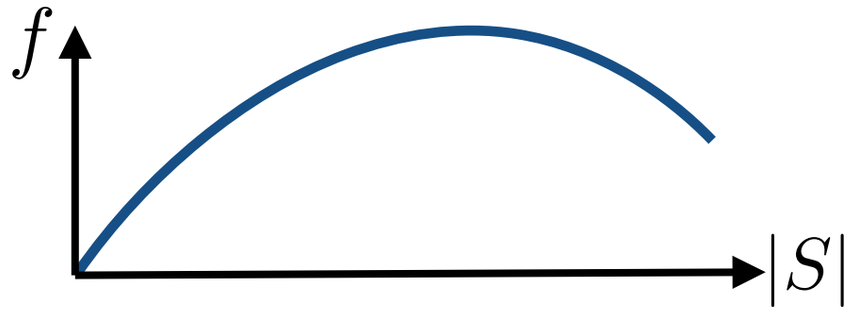
2020

"Approximability of monotone submodular function maximization under cardinality and matroid constraints in the streaming model ", C. Huang, N. Kakimura, S. Mauras, Y. Yoshida

Cardinality Constrained Streaming Submodular Maximization

- When the utility functions f is non-monotone submodular

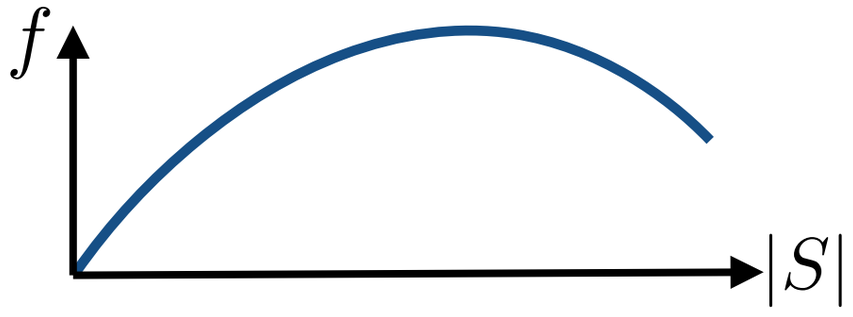
$$S^* = \arg\max_{|S| \leq k} f(S)$$



Cardinality Constrained Streaming Submodular Maximization

- When the utility functions f is non-monotone submodular

$$S^* = \arg\max_{|S| \leq k} f(S)$$

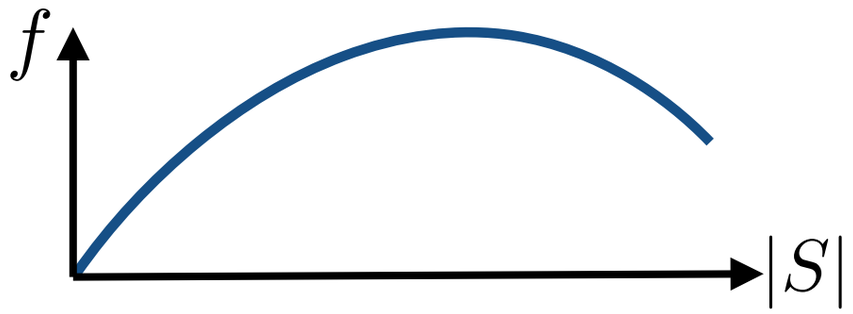


- Three ingredients
  - Thresholding
  - Guessing
  - Random partition

"Optimal Streaming Algorithms for Submodular Maximization with Cardinality Constraints", Alaluf, Ene, Feldma, Nguyen Suh, 2020

Cardinality Constrained Streaming Submodular Maximization

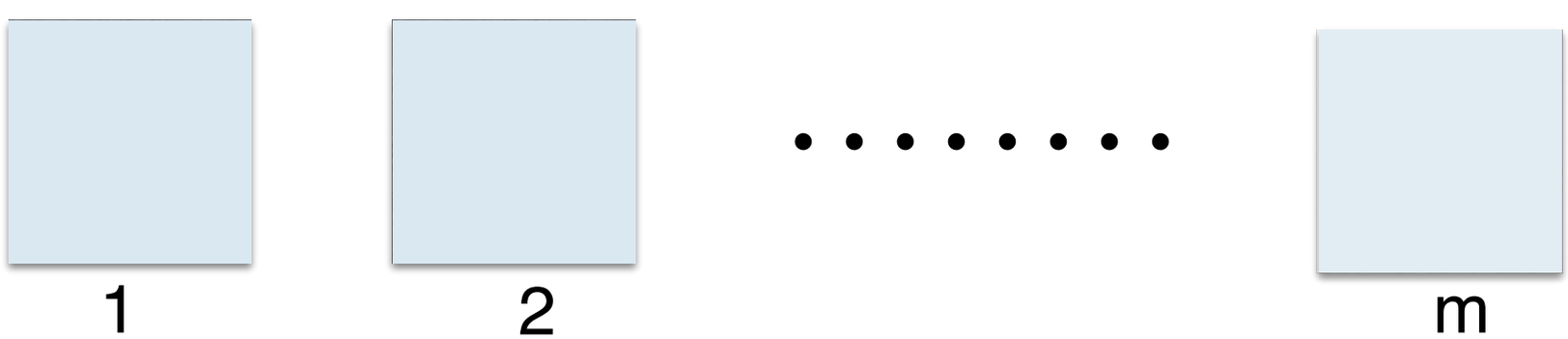
$$S^* = \arg\max_{|S| \leq k} f(S)$$



building block

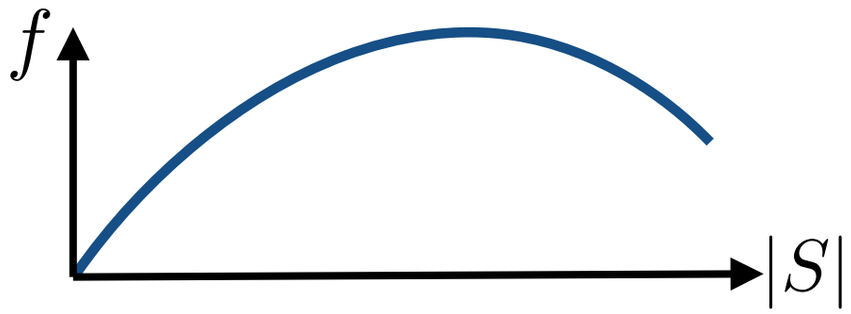
Data Stream

Marginal Gain



Cardinality Constrained Streaming Submodular Maximization

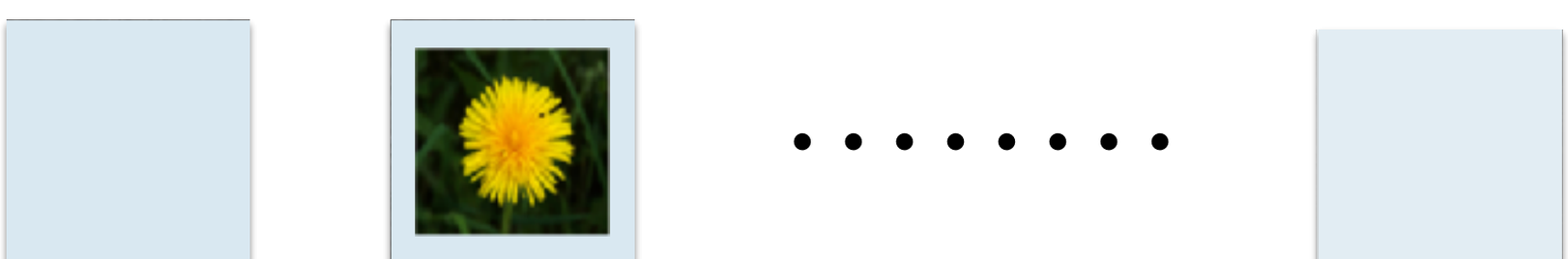
$$S^* = \arg\max_{|S| \leq k} f(S)$$



building block

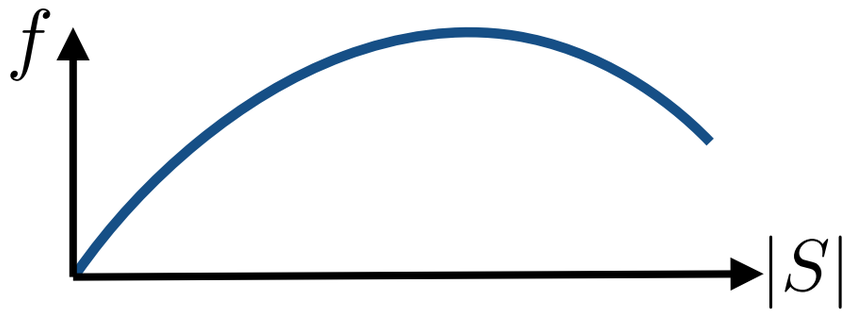
Data Stream

Marginal Gain



Cardinality Constrained Streaming Submodular Maximization

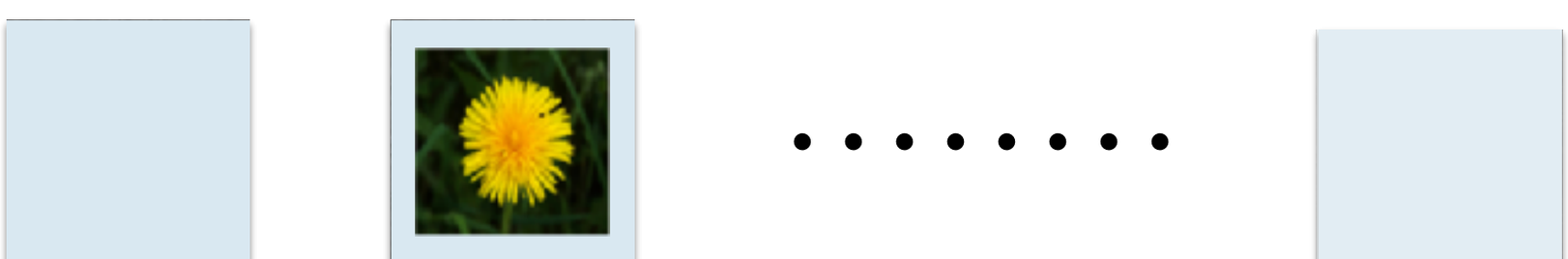
$$S^* = \arg\max_{|S| \leq k} f(S)$$



building block

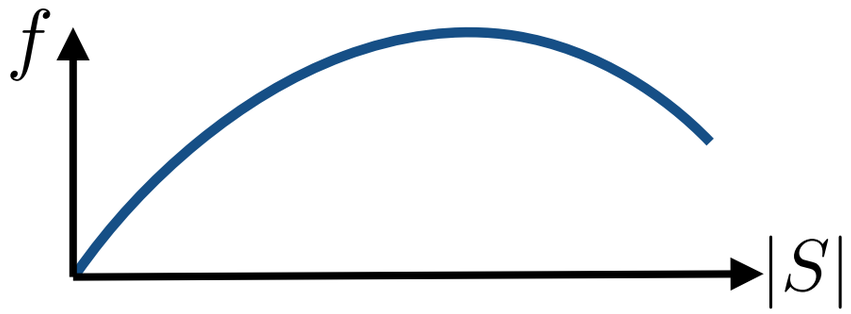
Data Stream

Marginal Gain



Cardinality Constrained Streaming Submodular Maximization

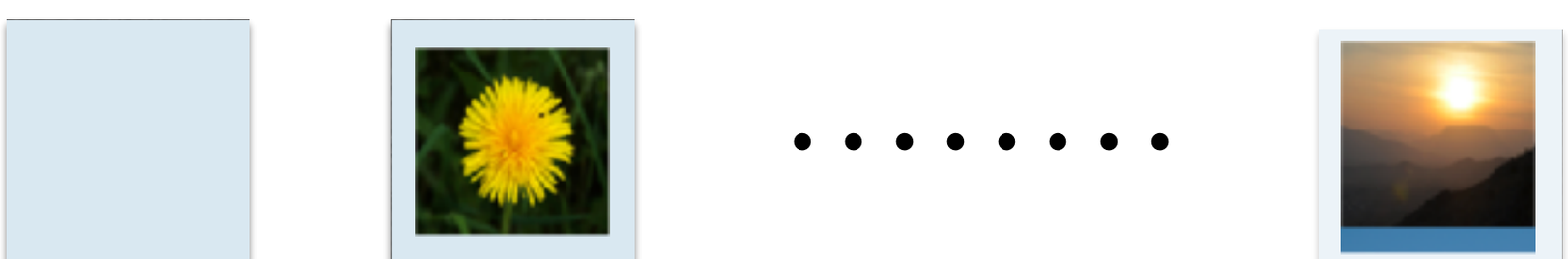
$$S^* = \arg\max_{|S| \leq k} f(S)$$



building block

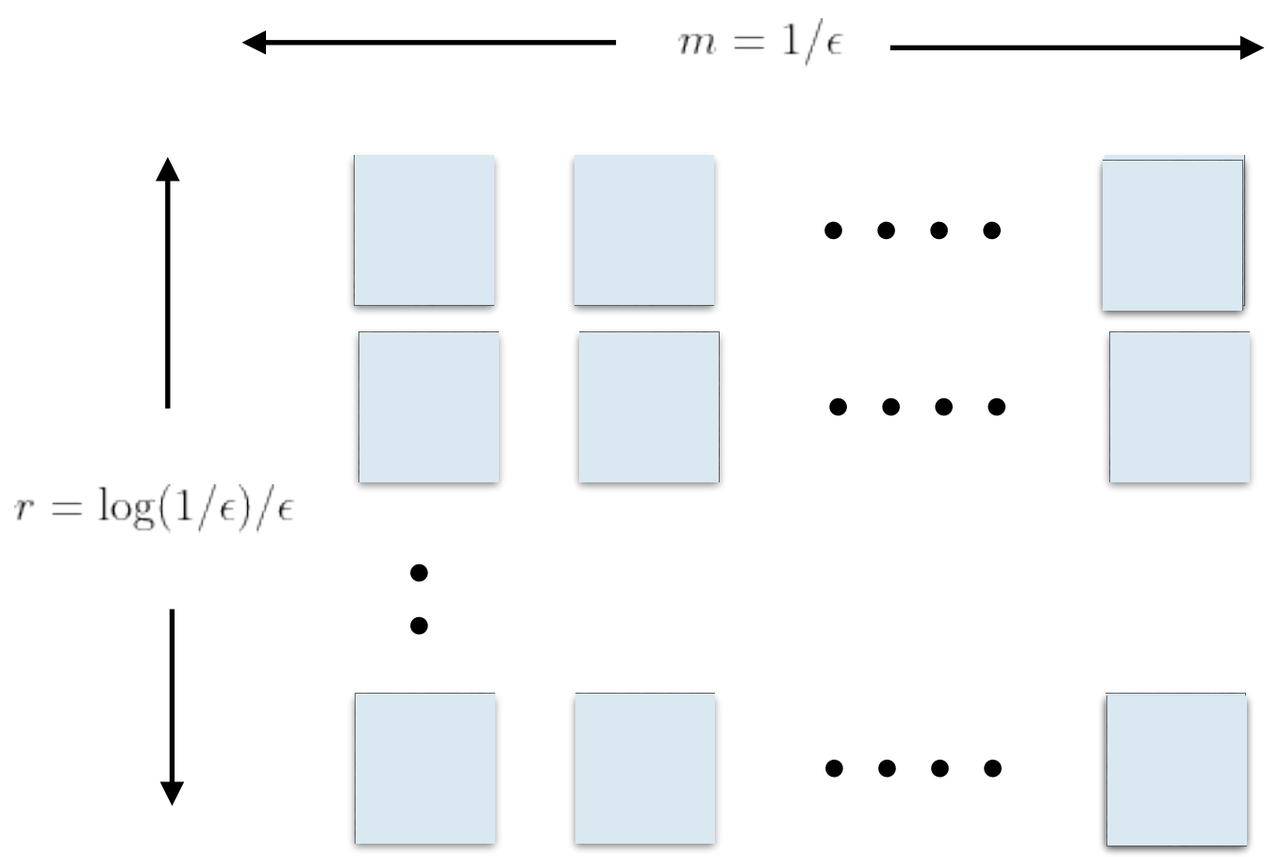
Data Stream

Marginal Gain



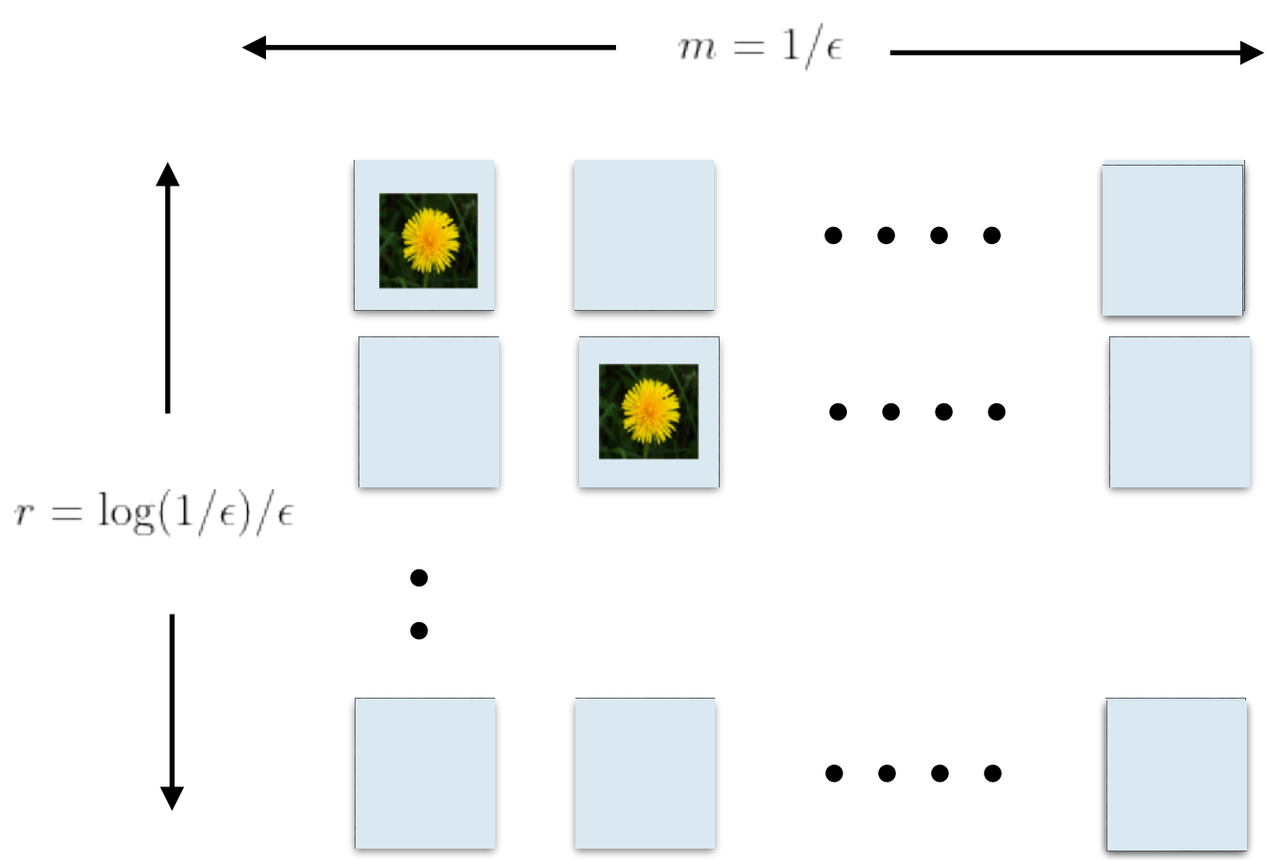
Cardinality Constrained Streaming Submodular Maximization

- Partition the data stream into $m = \frac{1}{\epsilon}$ parts.
- Run $r = \frac{\log(1/\epsilon)}{\epsilon}$ instances of the building block in parallel



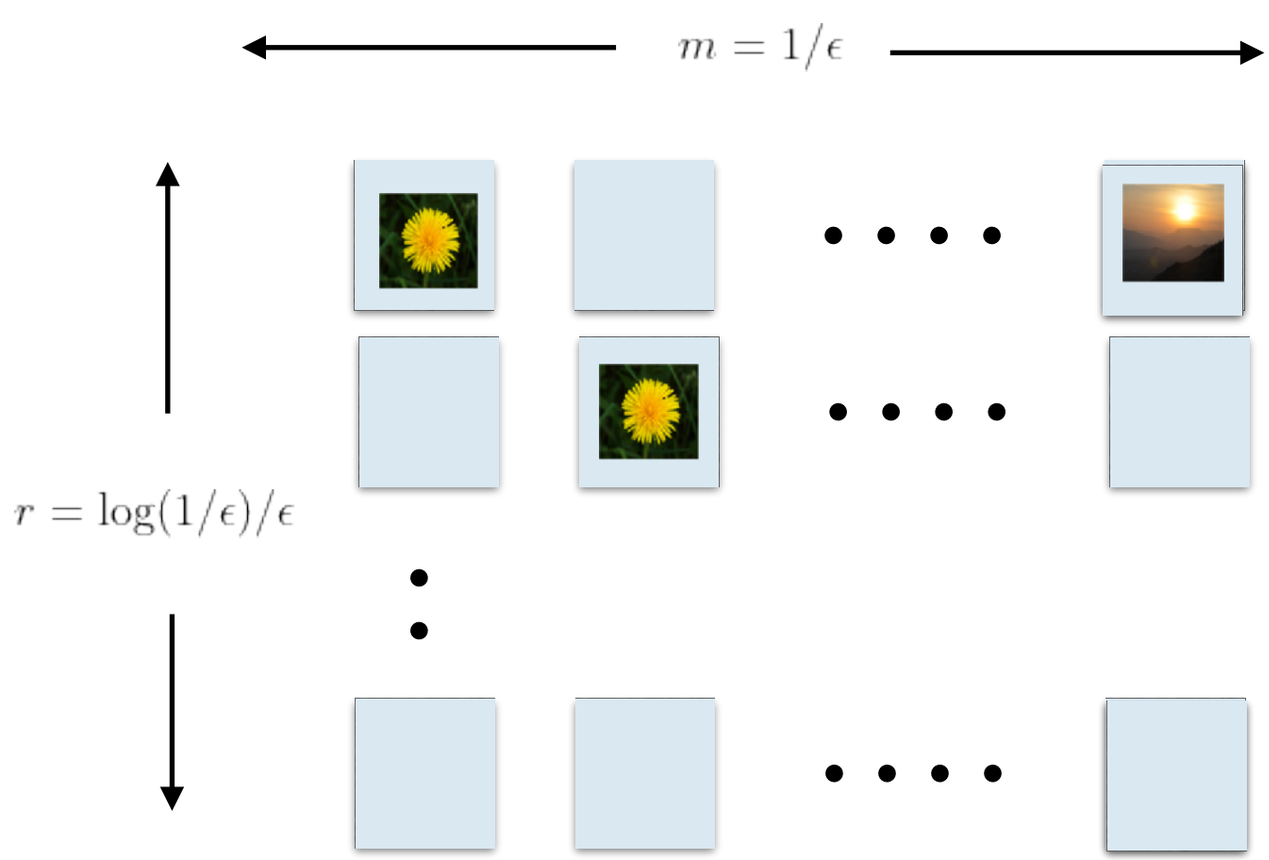
Cardinality Constrained Streaming Submodular Maximization

- Partition the data stream into $m = \frac{1}{\epsilon}$ parts.
- Run $r = \frac{\log(1/\epsilon)}{\epsilon}$ instances of the building block in parallel



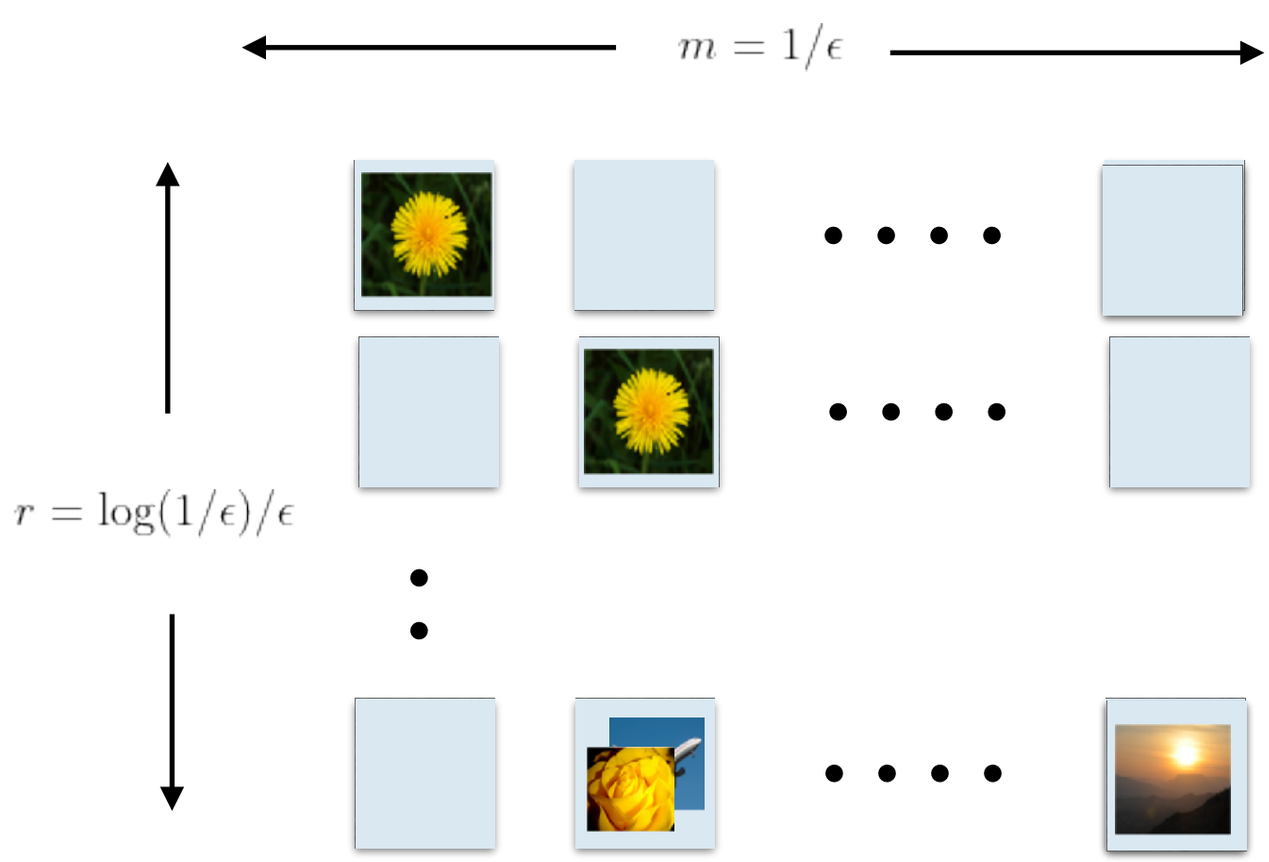
Cardinality Constrained Streaming Submodular Maximization

- Partition the data stream into $m = \frac{1}{\epsilon}$ parts.
- Run $r = \frac{\log(1/\epsilon)}{\epsilon}$ instances of the building block in parallel



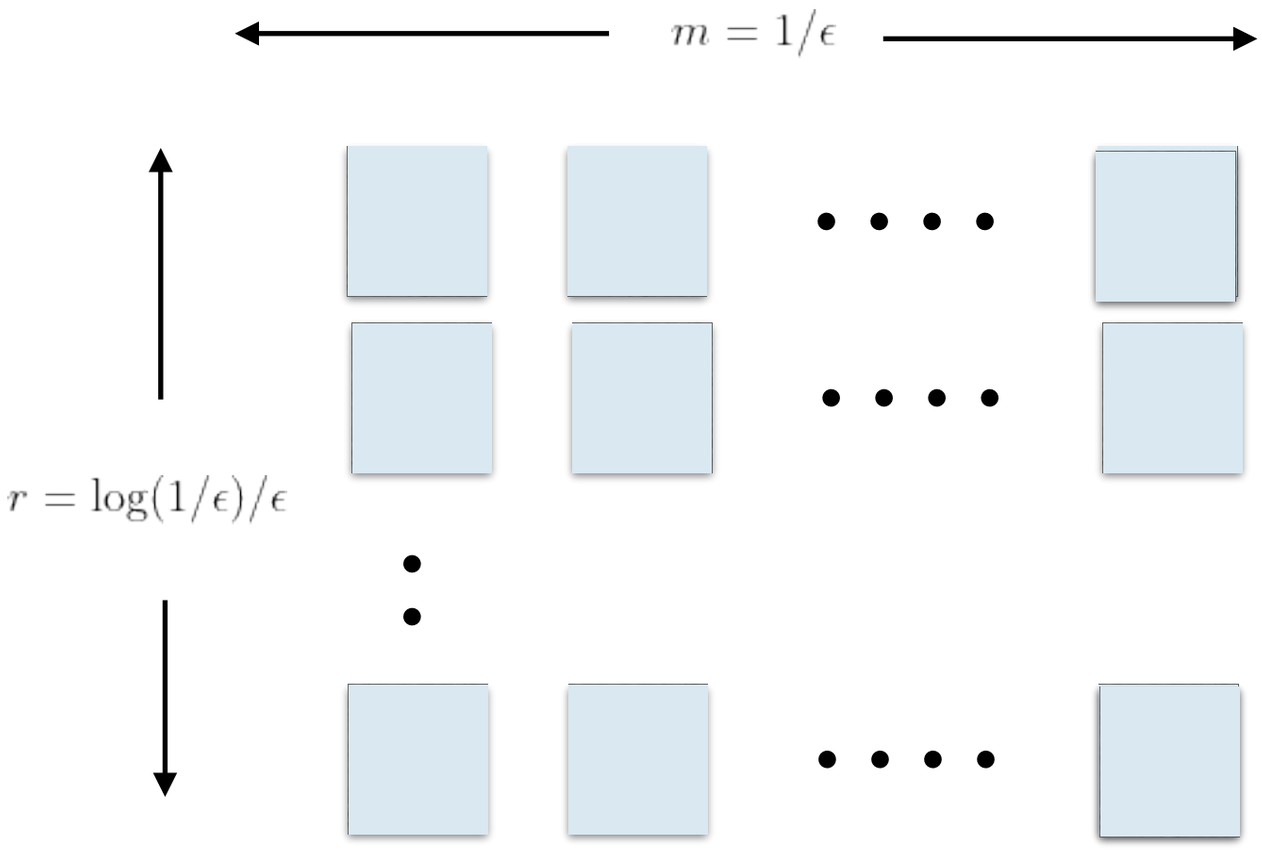
Cardinality Constrained Streaming Submodular Maximization

- Upon termination:
  - If any of the bocks has k elements, return it as the final solution.
  - Otherwise, take the union of all solutions and run an instance of the offline constrained submodular maximization.



Cardinality Constrained Streaming Submodular Maximization

- Upon termination:
  - If any of the bocks has k elements, return it as the final solution.
  - Otherwise, take the union of all solutions and run an instance of the offline constrained submodular maximization.

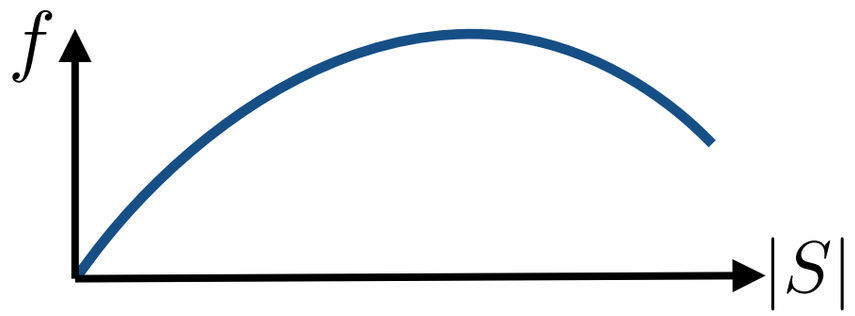


Offline Algorithm

Cardinality Constrained Streaming Submodular Maximization

- When the utility functions f is non-monotone submodular

$$S^* = \arg\max_{|S| \leq k} f(S)$$



[Alaluf, Ene, Feldma, Nguyen Suh, 2020]

For non-monotone submodular functions, and subject to a cardinality constraint, there is a one-pass streaming algorithm that achieves

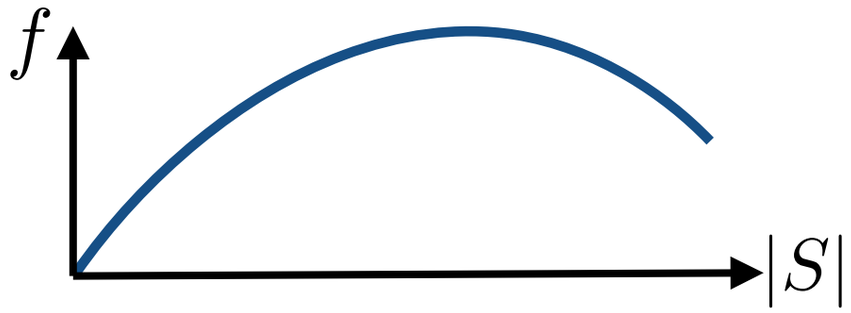
$$\mathbb{E}[f(S)] \geq \left(\frac{\alpha}{1+\alpha} - \epsilon\right) \mathrm{OPT}$$

where α is the approximation guarantee of the best offline algorithm.

"Optimal Streaming Algorithms for Submodular Maximization with Cardinality Constraints", 2020

Non-Monotone Streaming: Cardinality Constraint

$$S^* = \arg\max_{|S| \leq k} f(S)$$



Optimal Streaming Algorithms for Submodular Maximization with Cardinality Constraints

2020

Naor Alaluf* Alina Ene† Moran Feldman‡ Huy L. Nguyen§ Andrew Suh¶

$\alpha = 0.385$

?

Hard

0.277

0.491

Submodular Maximization by Simulated Annealing

Shayan Oveis Gharan* Jan Vondrák†

2010

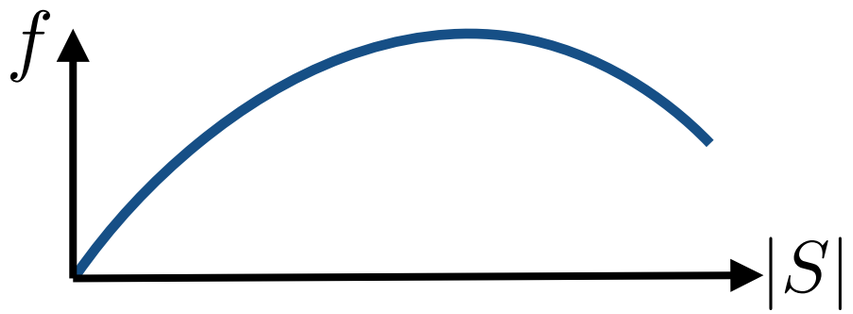
Streaming Non-Monotone Submodular Maximization

- Consider the following problem where f is submodular:

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

constraints

- When the utility functions f is non-monotone submodular



- and constraints imposed by the application

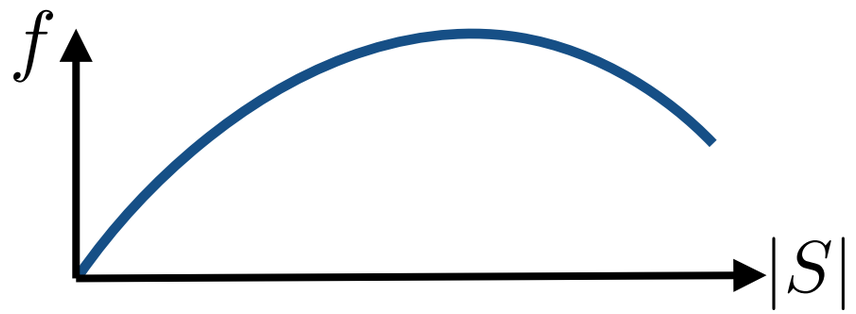
Streaming Non-Monotone Submodular Maximization

- Consider the following problem where f is submodular:

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

constraints

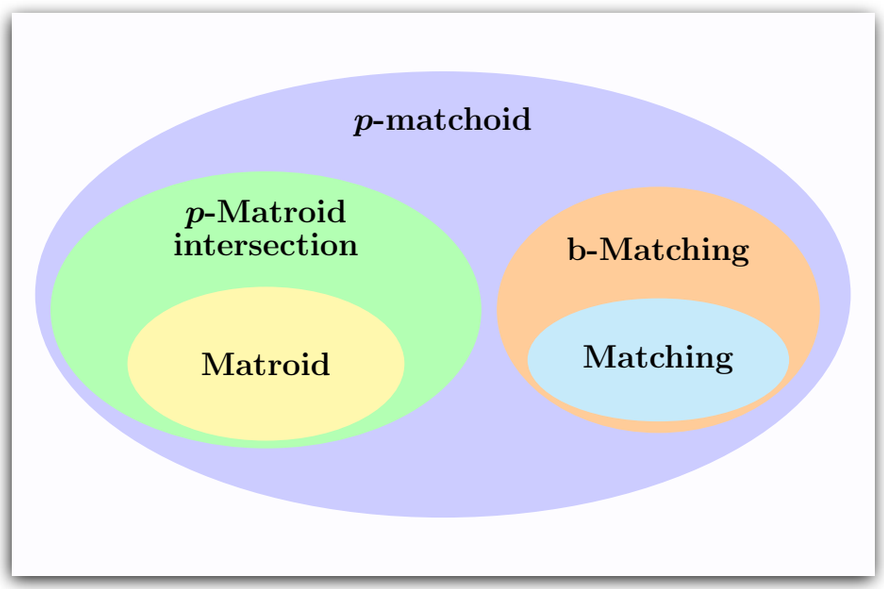
- When the utility functions f is non-monotone submodular



- and constraints imposed by the application

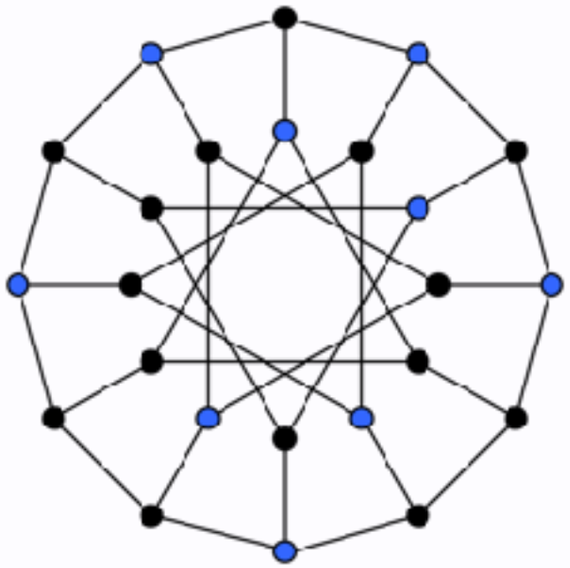
p-matchoid: a set system $(V, \mathcal{I})$ where there exist m matroids $(V_i, \mathcal{I}_i)$ such that every element of V appears in the ground set of at most p matroids and

$$\mathcal{I} = \left\{ S \subseteq 2^V | \forall_{1 \leq i \leq m} S \cap V_i \in \mathcal{I}_i \right\}$$



P-extendible

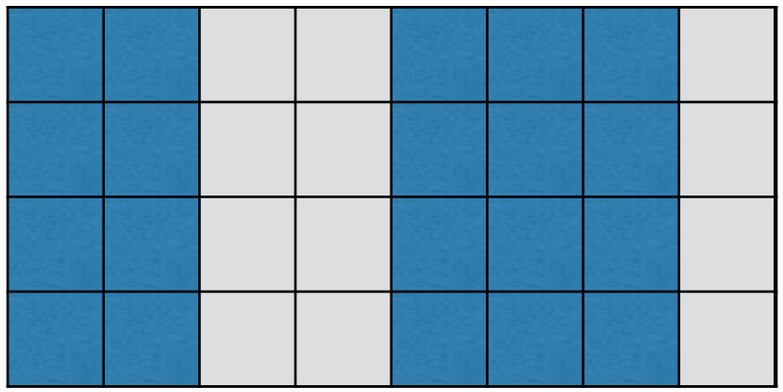
Independent Set $p = d_{\max}$



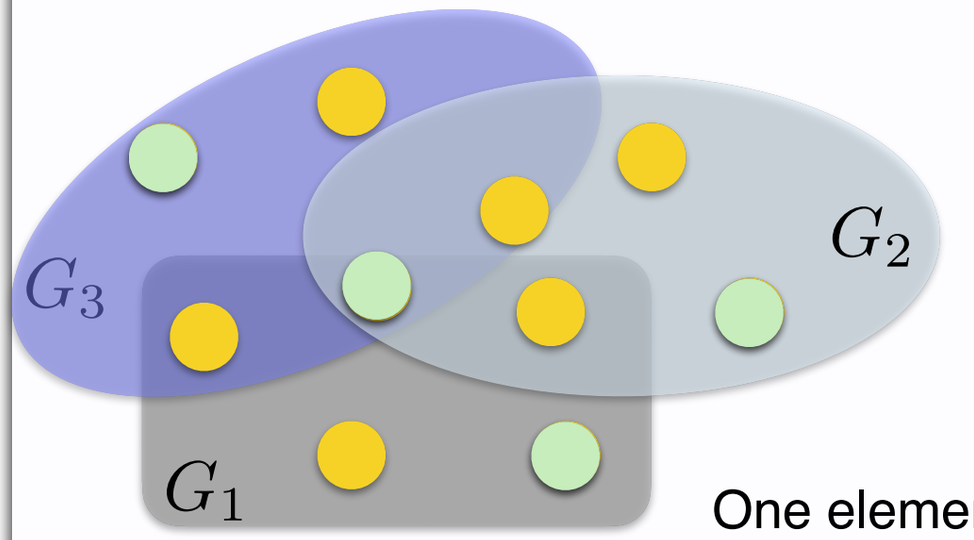
Matroid $p = 1$

$\forall A \subset B$ if $B \in \mathcal{I} \Rightarrow A \in \mathcal{I}$

$A, B \in \mathcal{I}$ & $|A| < |B| \Rightarrow \exists v \in B$ s.t. $A \cup \{v\} \in \mathcal{I}$

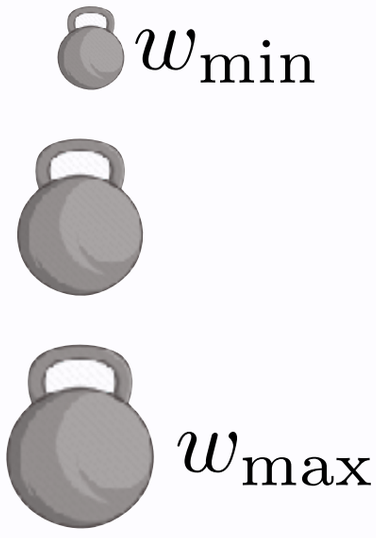


Intersection of k matroids $p = k$



One element from each group

Knapsack $p = \left\lceil \frac{w_{\max}}{w_{\min}} \right\rceil$

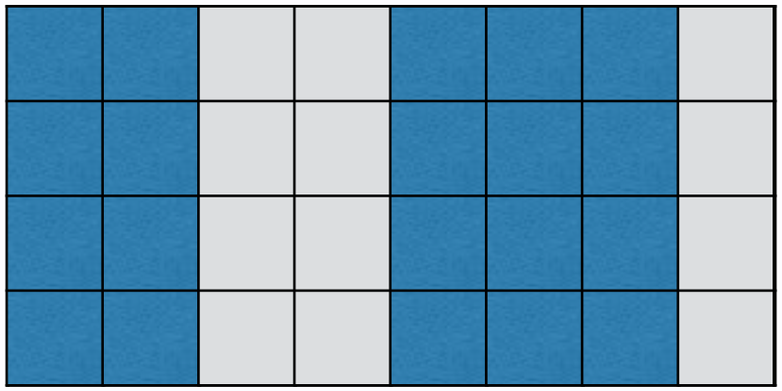


P-extendible => P-matchoid

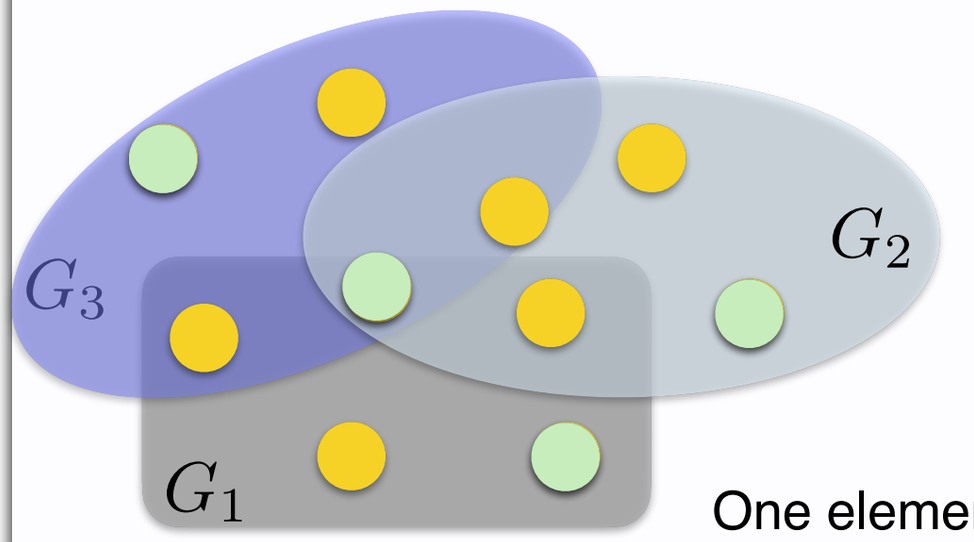
Matroid $p = 1$

$\forall A \subset B$ if $B \in \mathcal{I} \Rightarrow A \in \mathcal{I}$

$A, B \in \mathcal{I}$ & $|A| < |B| \Rightarrow \exists v \in B$ s.t. $A \cup \{v\} \in \mathcal{I}$



Intersection of k matroids $p = k$



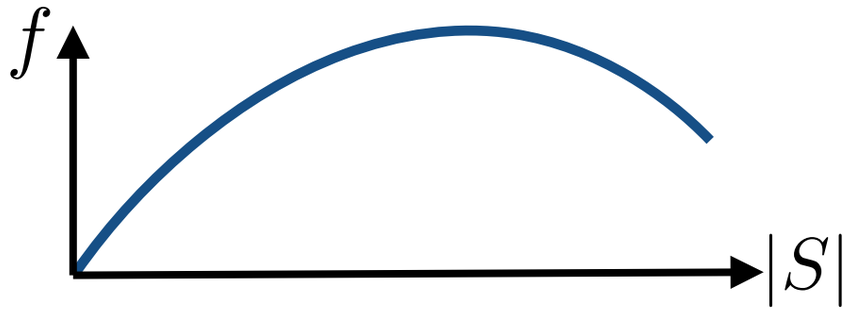
One element from each group

Cardinality Constrained Streaming Submodular Maximization

- When the utility functions f is non-monotone submodular

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

P-matchoid

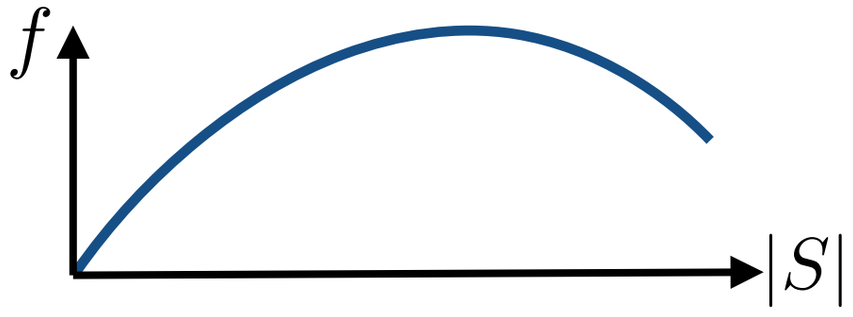


Cardinality Constrained Streaming Submodular Maximization

- When the utility functions f is non-monotone submodular

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

P-matchoid



- Three ingredients
  - Thresholding
  - Guessing
  - Buffer

"Streaming Algorithms for Submodular Function Maximization", Chekuri, Gupta, Quanrud, 2015.

Cardinality Constrained Streaming Submodular Maximization

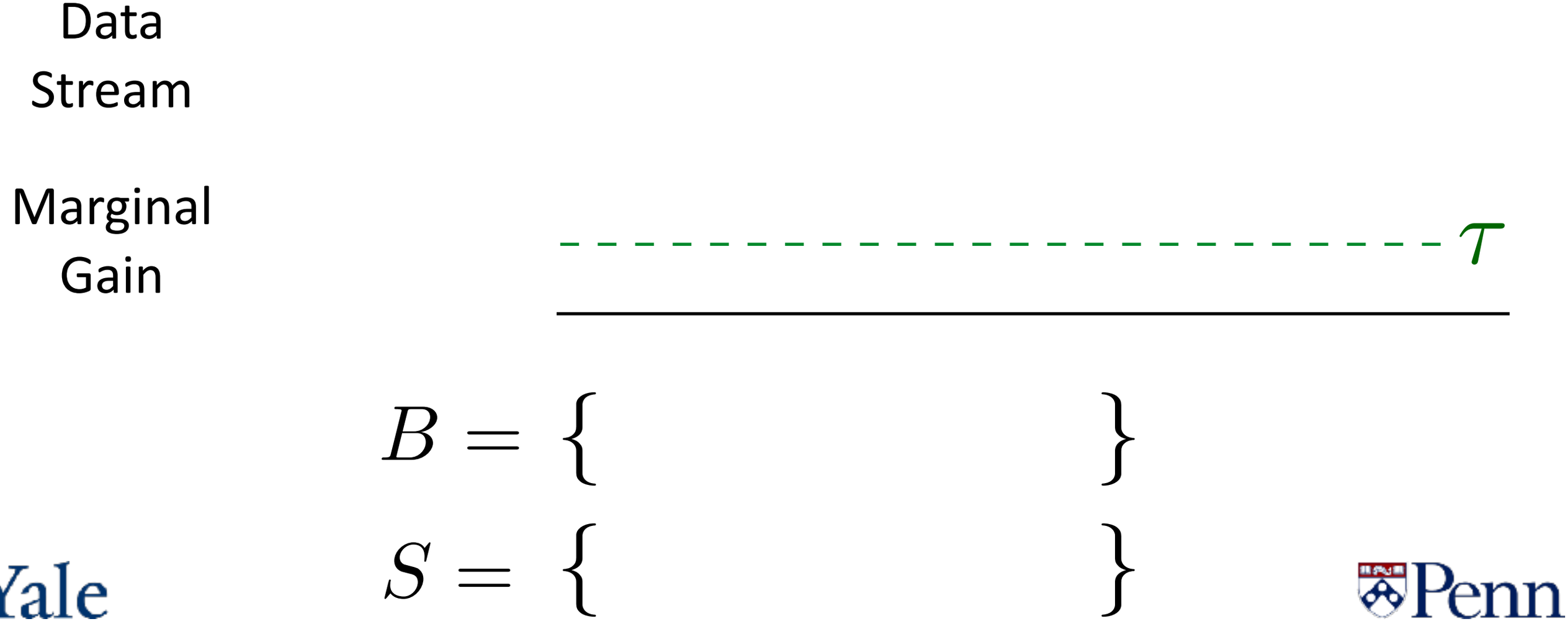
- For each item e in the stream:
  - If the marginal gain of e is greater than the threshold τ, add e to the buffer B.
  - If the buffer B becomes full:
    1. Select one item uniformly at random to move to solution set S.
    2. Filter remaining buffered items based on their new marginal gains.

Yale

Penn

Cardinality Constrained Streaming Submodular Maximization

- For each item e in the stream:
  - If the marginal gain of e is greater than the threshold τ, add e to the buffer B.
  - If the buffer B becomes full:
    1. Select one item uniformly at random to move to solution set S.
    2. Filter remaining buffered items based on their new marginal gains.



Yale

Penn

Cardinality Constrained Streaming Submodular Maximization

- For each item e in the stream:
  - If the marginal gain of e is greater than the threshold τ, add e to the buffer B.
  - If the buffer B becomes full:
    1. Select one item uniformly at random to move to solution set S.
    2. Filter remaining buffered items based on their new marginal gains.

Data Stream

Marginal Gain

Yale

Penn

Cardinality Constrained Streaming Submodular Maximization

- For each item e in the stream:
  - If the marginal gain of e is greater than the threshold τ, add e to the buffer B.
  - If the buffer B becomes full:
    1. Select one item uniformly at random to move to solution set S.
    2. Filter remaining buffered items based on their new marginal gains.

$$B = \{ \quad \quad \quad \}$$

$$S = \{ \quad \}$$

Yale

Penn

Cardinality Constrained Streaming Submodular Maximization

- For each item e in the stream:
  - If the marginal gain of e is greater than the threshold τ, add e to the buffer B.
  - If the buffer B becomes full:
    1. Select one item uniformly at random to move to solution set S.
    2. Filter remaining buffered items based on their new marginal gains.

$B = \{$   $\}$

$S = \{$  $\}$

Yale

Penn

Cardinality Constrained Streaming Submodular Maximization

- For each item e in the stream:
  - If the marginal gain of e is greater than the threshold τ, add e to the buffer B.
  - If the buffer B becomes full:
    1. Select one item uniformly at random to move to solution set S.
    2. Filter remaining buffered items based on their new marginal gains.

New Marginal Gains

τ

$B = \{$  $\}$

$S = \{$  $\}$

Yale

Penn

Cardinality Constrained Streaming Submodular Maximization

- For each item e in the stream:
  - If the marginal gain of e is greater than the threshold τ, add e to the buffer B.
  - If the buffer B becomes full:
    1. Select one item uniformly at random to move to solution set S.
    2. Filter remaining buffered items based on their new marginal gains.

Data Stream

Marginal Gain

τ

$B = \{$  $\}$

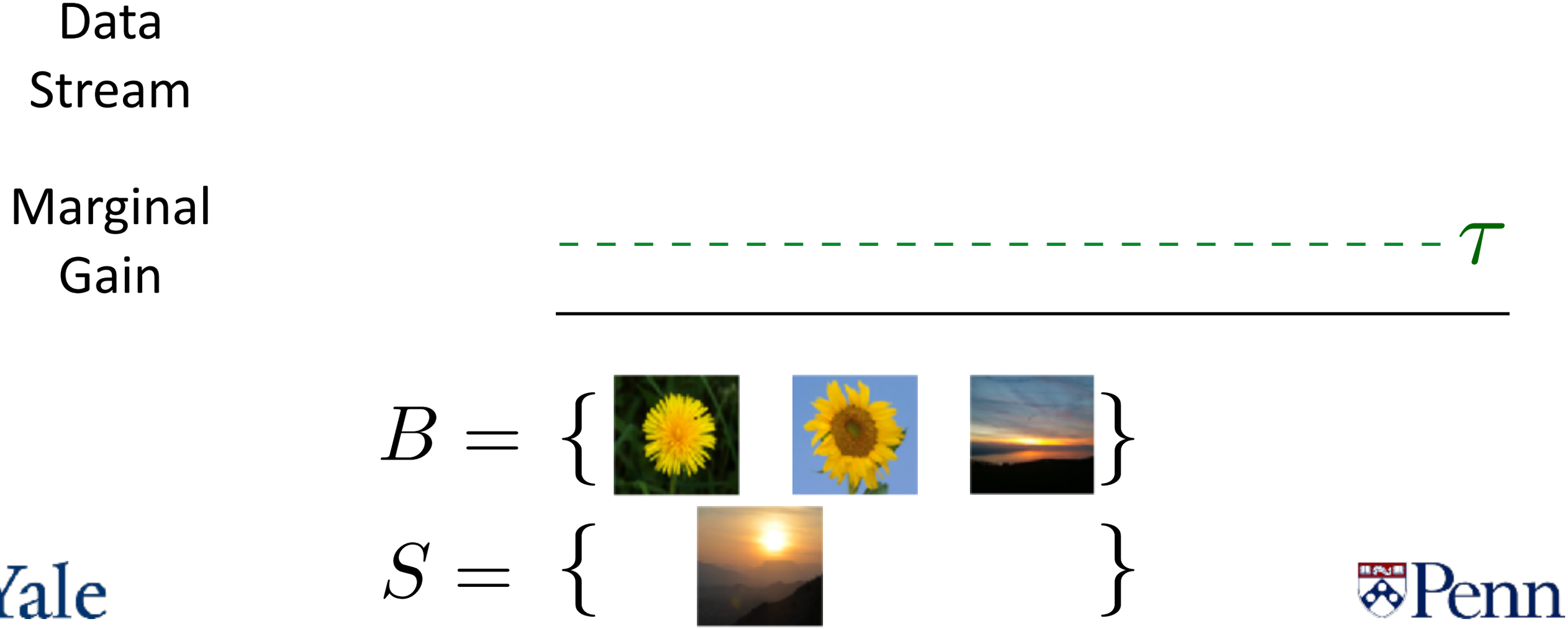
$S = \{$  $\}$

Yale

Penn

Cardinality Constrained Streaming Submodular Maximization

- For each item e in the stream:
  - If the marginal gain of e is greater than the threshold τ, add e to the buffer B.
  - If the buffer B becomes full:
    1. Select one item uniformly at random to move to solution set S.
    2. Filter remaining buffered items based on their new marginal gains.



Yale

Penn

Cardinality Constrained Streaming Submodular Maximization

- For each item e in the stream:
  - If the marginal gain of e is greater than the threshold τ, add e to the buffer B.
  - If the buffer B becomes full:
    1. Select one item uniformly at random to move to solution set S.
    2. Filter remaining buffered items based on their new marginal gains.

$B = \{$  $\}$

$S = \{$  $\}$

Yale

Penn

Cardinality Constrained Streaming Submodular Maximization

- For each item e in the stream:
  - If the marginal gain of e is greater than the threshold τ, add e to the buffer B.
  - If the buffer B becomes full:
    1. Select one item uniformly at random to move to solution set S.
    2. Filter remaining buffered items based on their new marginal gains.

$B = \{$   $\}$

$S = \{$   $\}$

Yale

Penn

Randomized Streaming Greedy

- For each item e in the stream:
  - If the marginal gain of e is greater than the threshold τ, add e to the buffer B.
  - If the buffer B becomes full:
    1. Select one item uniformly at random to move to solution set S.
    2. Filter remaining buffered items based on their new marginal gains.

[Chekuri, Gupta, Quanrud]

For non-negative submodular functions under a p-matchoid constraint, Streaming-Greedy with an $O(k \log k/\epsilon^2)$ memory gives constant factor approximation:

- if f is **monotone**:

$$\mathbb{E}[f(S_{\text{Streaming-greedy}})] \geq \frac{1}{4p}\text{OPT}$$

- if f is **non-monotone**:

$$\mathbb{E}[f(S_{\text{Streaming-greedy}})] \geq \frac{1-\epsilon}{5p+2+1/p}\text{OPT}$$

"Streaming Algorithms for Submodular Function Maximization", 2015

Yale

Penn

The Sample-Streaming Algorithm

- For each item u in the stream:
  - Keep u with probability $q = \frac{1}{p+\sqrt{p(p+1)}+1}$
  - If we kept u: swap u into S so that we maintain feasibility and check that the marginal gain is large enough.

The Sample-Streaming Algorithm

- For each item u in the stream:
  - Keep u with probability $q = \frac{1}{p+\sqrt{p(p+1)}+1}$
  - If we kept u: swap u into S so that we maintain feasibility and check that the marginal gain is large enough.

Data Stream

$$S = \{$$    $$\}$$

Yale

Penn

The Sample-Streaming Algorithm

- For each item u in the stream:
  - Keep u with probability $q = \frac{1}{p+\sqrt{p(p+1)}+1}$
  - If we kept u: swap u into S so that we maintain feasibility and check that the marginal gain is large enough.

Data Stream

$S = \{$    $\}$

Yale

Penn

The Sample-Streaming Algorithm

- For each item u in the stream:
  - Keep u with probability $q = \frac{1}{p+\sqrt{p(p+1)}+1}$
  - If we kept u: swap u into S so that we maintain feasibility and check that the marginal gain is large enough.

Data Stream

$$S = \{$$    $$\}$$

Yale

Penn

Sample-Streaming: Guarantee

[Feldman, Karbasi, Kazemi]

For non-negative submodular functions under a p-matchoid constraint, Sample-Streaming with an $O(k)$ memory gives constant factor approximation using only $O(km/p)$ function evaluation per element:

- if f is **monotone**:

$$\mathbb{E}\left[f\left(S_{\text{SAMPLE-STREAMING}}\right)\right] \geq \frac{1}{4p}\text{OPT}$$

- if f is **non-monotone**:

$$\mathbb{E}[f(S_{\text{SAMPLE-STREAMING}})] \geq \frac{1}{4p+2-o(1)}\text{OPT}$$

"Do Less, Get More: Streaming Submodular Maximization with Subsampling", 2018

Summary

"Streaming non-monotone submodular maximization: Personalized video summarization on the fly", Mirzasoleiman, Jegelka, Krause, 2018

Sample-Streaming: Guarantee

[Feldman, Karbasi, Kazemi]

For non-negative submodular functions under a p-matchoid constraint, Sample-Streaming with an $O(k)$ memory gives constant factor approximation using only $O(km/p)$ function evaluation per element:

- if f is **monotone**:

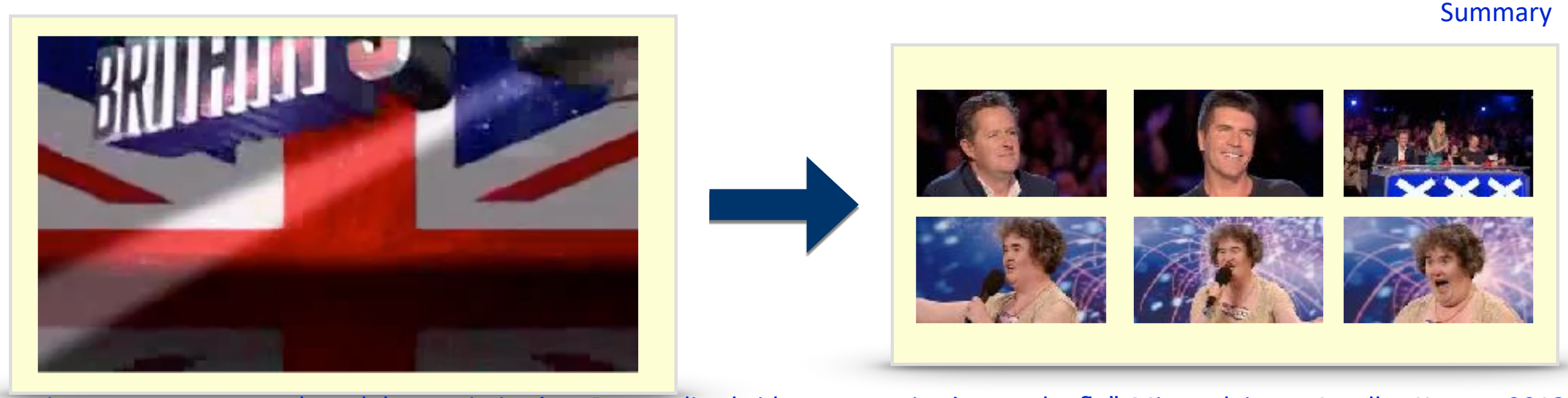
$$\mathbb{E}\left[f\left(S_{\text{SAMPLE-STREAMING}}\right)\right] \geq \frac{1}{4p}\text{OPT}$$

- if f is **non-monotone**:

$$\mathbb{E}[f(S_{\text{SAMPLE-STREAMING}})] \geq \frac{1}{4p+2-o(1)}\text{OPT}$$

“Do Less, Get More: Streaming Submodular Maximization with Subsampling”, 2018

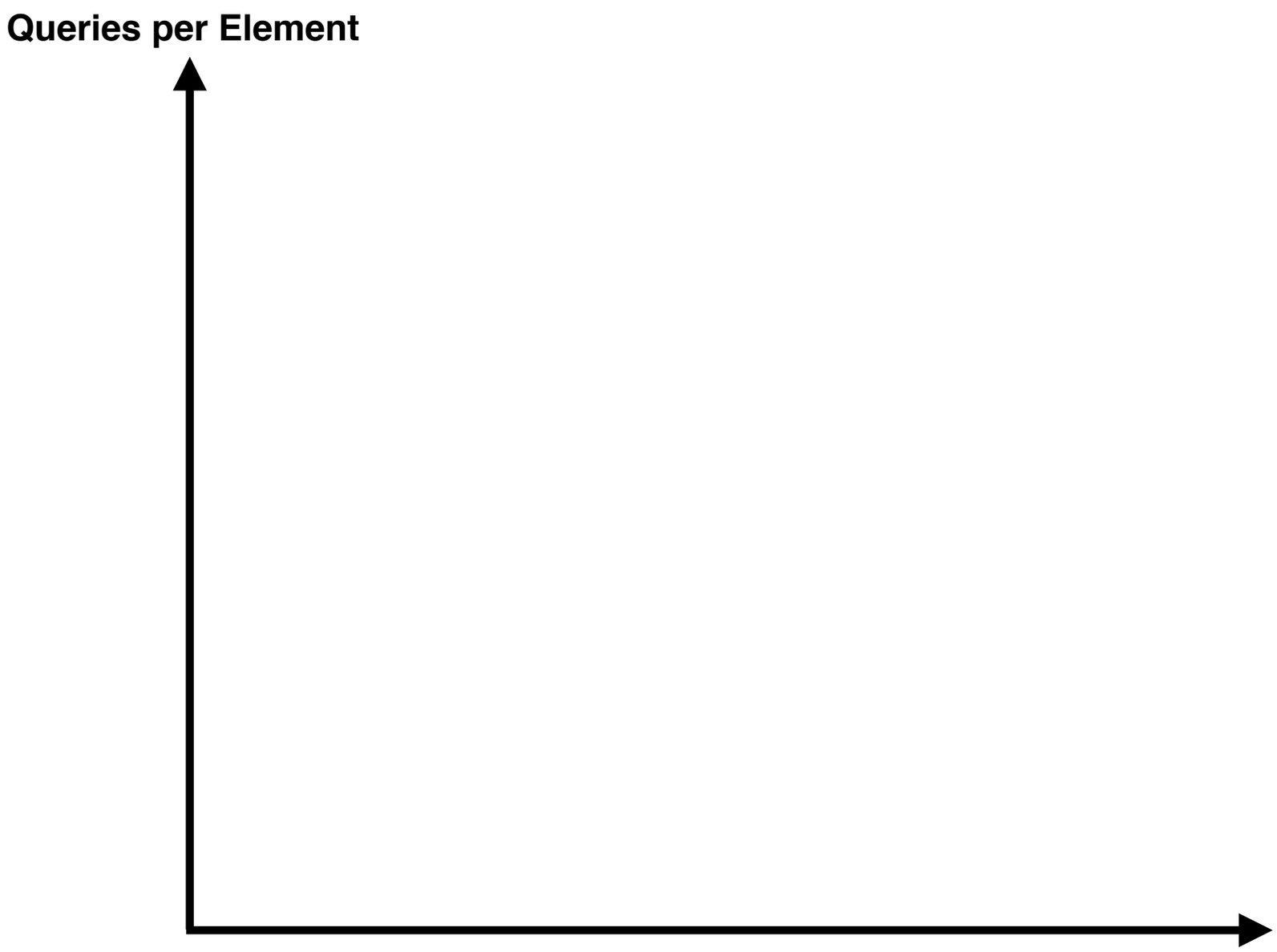
Summary



“Streaming non-monotone submodular maximization: Personalized video summarization on the fly”, Mirzasoleiman, Jegelka, Krause, 2018

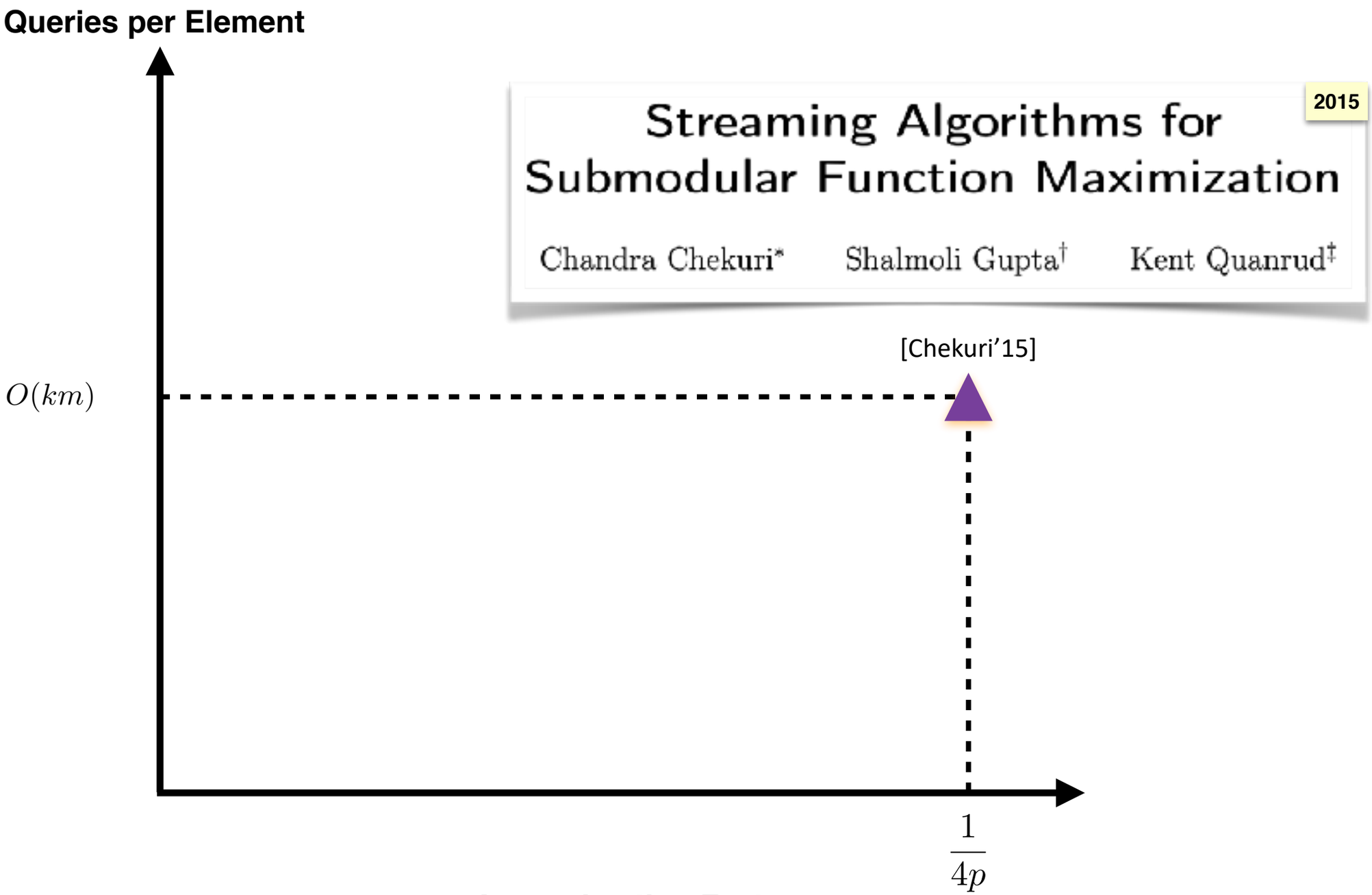
Streaming Algorithms Subject to p-matchoid

Monotone



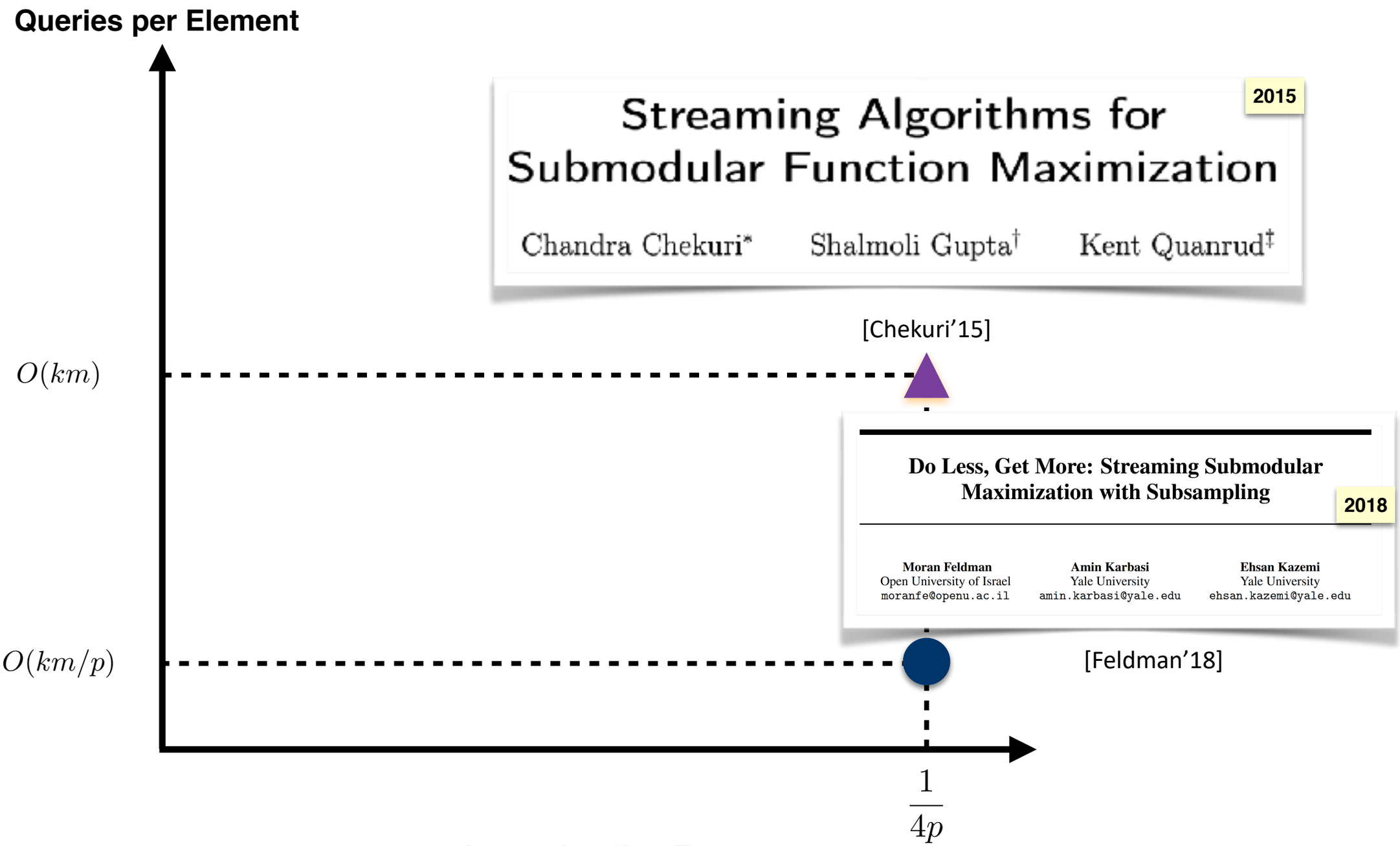
Streaming Algorithms Subject to p-matchoid

Monotone



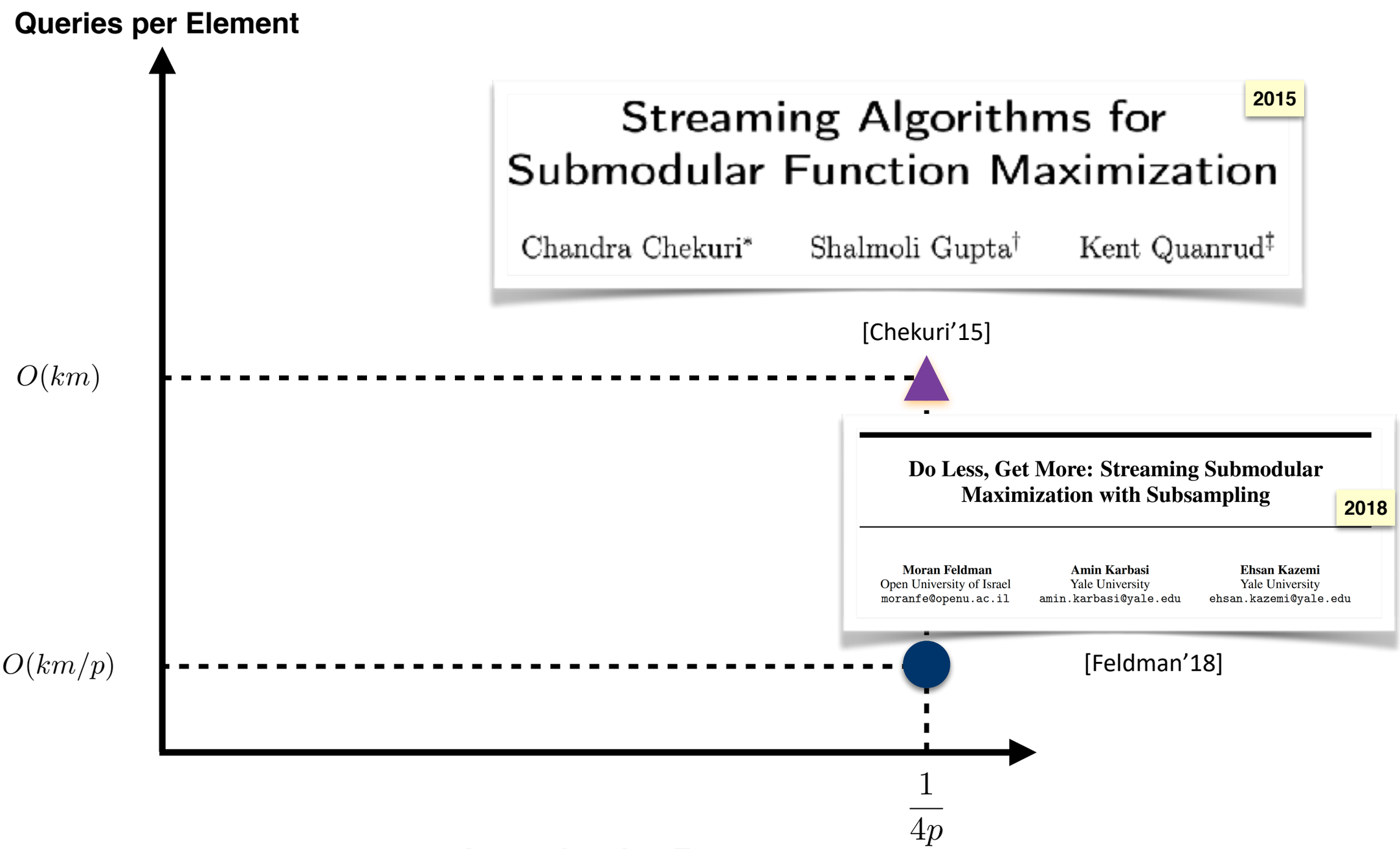
Streaming Algorithms Subject to p-matchoid

Monotone



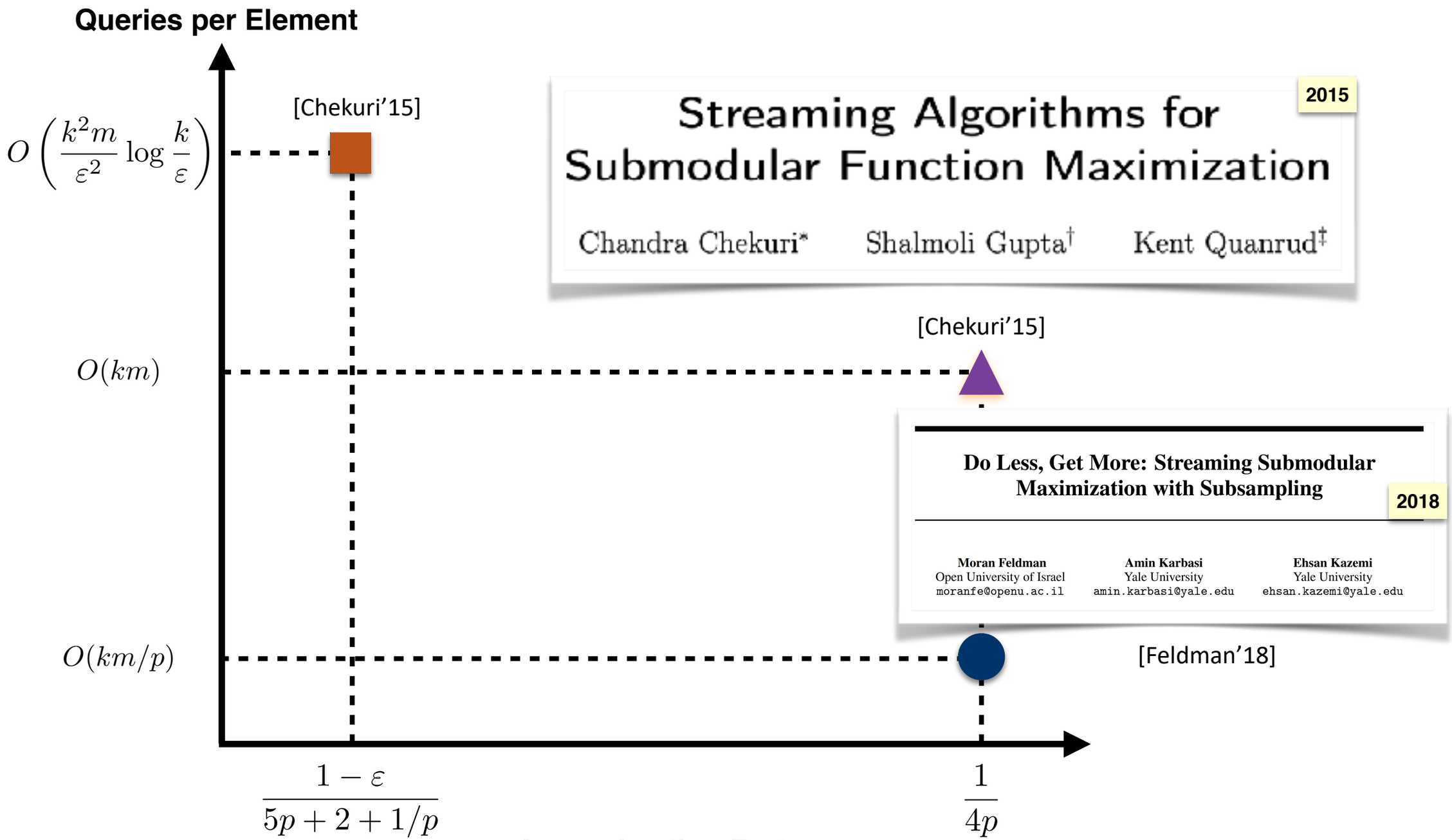
Streaming Algorithms Subject to p-matchoid

Monotone → Non-montone



Streaming Algorithms Subject to p-matchoid

Monotone → Non-montone



2015

Streaming Algorithms for Submodular Function Maximization

Chandra Chekuri* Shalmoli Gupta† Kent Quanrud‡

2018

Do Less, Get More: Streaming Submodular Maximization with Subsampling

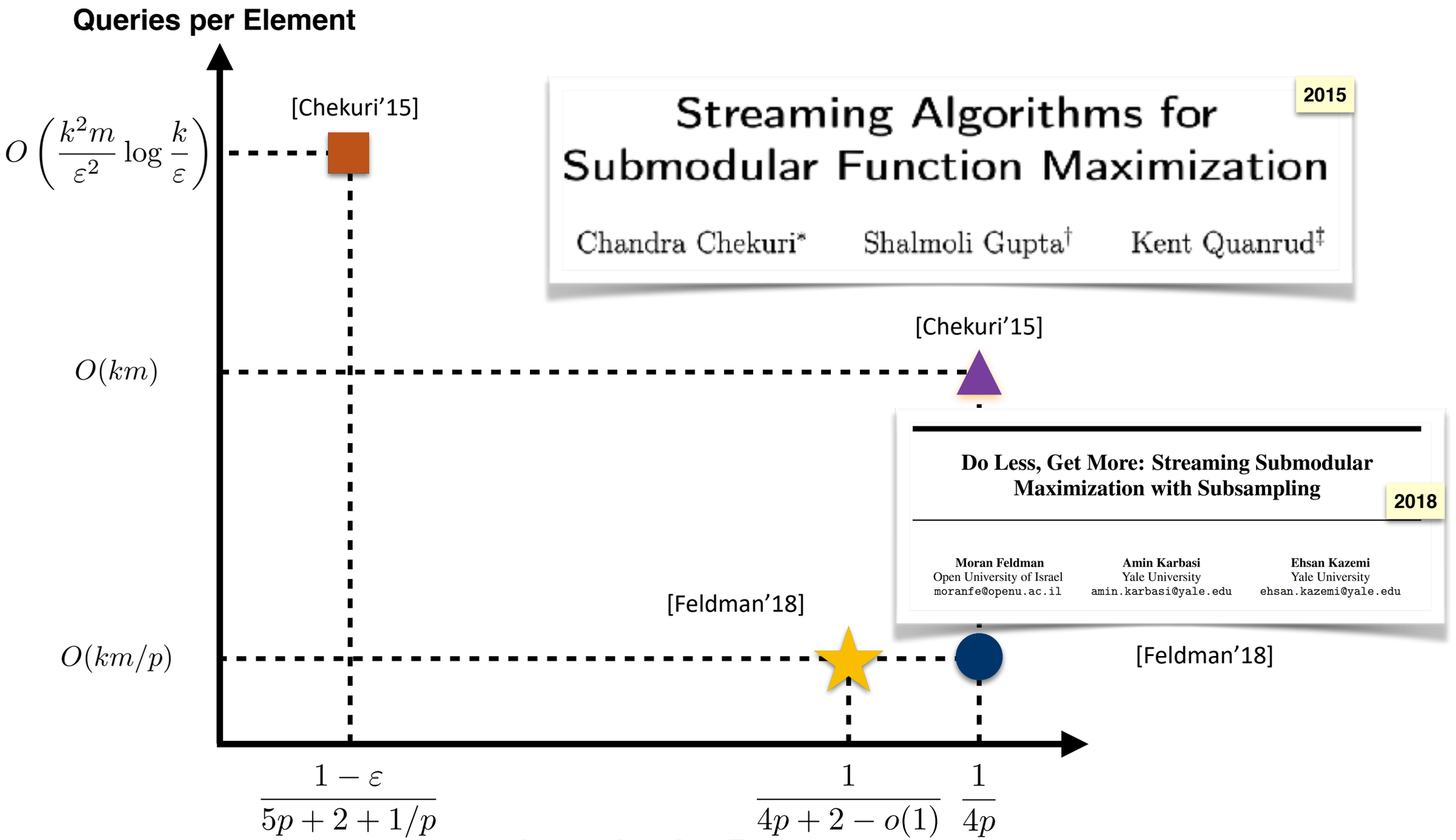
Moran Feldman
Open University of Israel
moranfe@openu.ac.il

Amin Karbasi
Yale University
amin.karbasi@yale.edu

Ehsan Kazemi
Yale University
ehsan.kazemi@yale.edu

Streaming Algorithms Subject to p-matchoid

Monotone → Non-montone



Streaming Algorithms for Submodular Function Maximization

2015

Chandra Chekuri* Shalmoli Gupta† Kent Quanrud‡

Do Less, Get More: Streaming Submodular Maximization with Subsampling

2018

Moran Feldman
Open University of Israel
moranfe@openu.ac.il

Amin Karbasi
Yale University
amin.karbasi@yale.edu

Ehsan Kazemi
Yale University
ehsan.kazemi@yale.edu

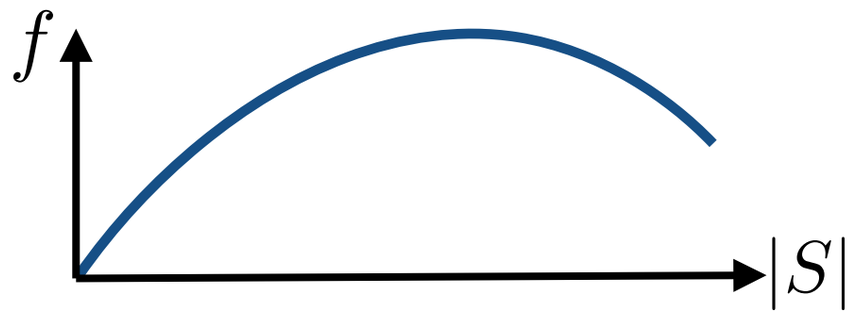
Constrained Non-Monotone Submodular Maximization

- Consider the following problem where f is submodular:

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

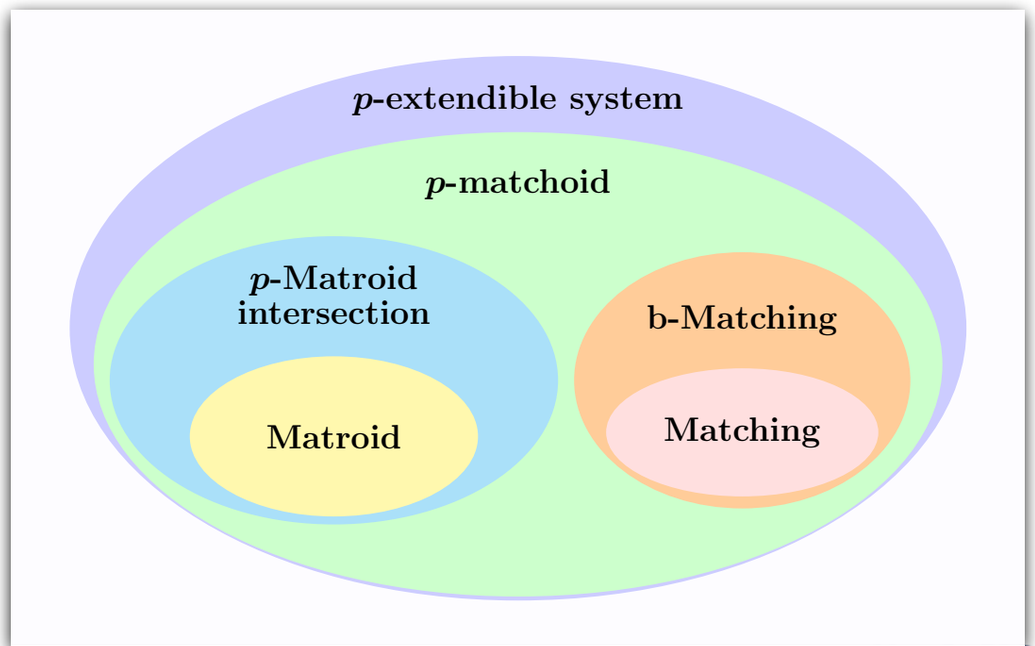
constraints

- When the utility functions f is non-monotone submodular



- and constraints imposed by the application

p-extendible system: adding an element e to a feasible set B requires the removal of at most p elements in order to keep the resulting set feasible.



Open Question: K-Extendible and K-System

Streaming Submodular Maximization under a k-Set System Constraint

Ran Haba* Ehsan Kazemi† Moran Feldman† Amin Karbasi§

2020

$$O\left(\frac{1}{p \log p}\right)$$

Hard

$$\frac{1}{p + 1/2 - \varepsilon}$$

Greed Is Good: Near-Optimal Submodular Maximization via Greedy Optimization

Moran Feldman MORANFE@OPENU.AC.IL
Department of Mathematics and Computer Science
The Open University of Israel

Christopher Harshaw CHRISTOPHER.HARSHAW@YALE.EDU
Department of Computer Science
Yale Institute for Network Science
Yale University

Amin Karbasi AMIN.KARBASI@YALE.EDU
Department of Electrical Engineering and Computer Science
Yale Institute for Network Science
Yale University

2017

Open Question: K-Extendible and K-System

Streaming Submodular Maximization under a k-Set System Constraint

Ran Haba* Ehsan Kazemi† Moran Feldman† Amin Karbasi§

2020

$$O\left(\frac{1}{p \log p}\right)$$

?

Hard

$$\frac{1}{p + 1/2 - \varepsilon}$$

Greed Is Good: Near-Optimal Submodular Maximization via Greedy Optimization

Moran Feldman MORANFE@OPENU.AC.IL
Department of Mathematics and Computer Science
The Open University of Israel

Christopher Harshaw CHRISTOPHER.HARSHAW@YALE.EDU
Department of Computer Science
Yale Institute for Network Science
Yale University

Amin Karbasi AMIN.KARBASI@YALE.EDU
Department of Electrical Engineering and Computer Science
Yale Institute for Network Science
Yale University

2017

Open Question: K-Extendible and K-System

Streaming Submodular Maximization under a k-Set System Constraint

Ran Haba* Ehsan Kazemi† Moran Feldman† Amin Karbasi§

2020

$$O\left(\frac{1}{p \log p}\right)$$

?

Hard

$$O\left(\frac{1}{p}\right)?$$

$$\frac{1}{p + 1/2 - \varepsilon}$$

Greed Is Good: Near-Optimal Submodular Maximization via Greedy Optimization

Moran Feldman MORANFE@OPENU.AC.IL
Department of Mathematics and Computer Science
The Open University of Israel

Christopher Harshaw CHRISTOPHER.HARSHAW@YALE.EDU
Department of Computer Science
Yale Institute for Network Science
Yale University

Amin Karbasi AMIN.KARBASI@YALE.EDU
Department of Electrical Engineering and Computer Science
Yale Institute for Network Science
Yale University

2017

This Talk

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

Perfect Oracle

Discrete Methods

- **Centralized**
- **Streaming**
- **Distributed**

Imperfect Oracle

Continuous Methods

This Talk

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

Perfect Oracle

Discrete Methods

- **Centralized**
- **Streaming**
- **Distributed**

Imperfect Oracle

Continuous Methods

Scaling Up

- On massive data (45M users' logs) the greedy policies take a few days/ weeks to complete

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

Can we parallelize the greedy approach?

Setting

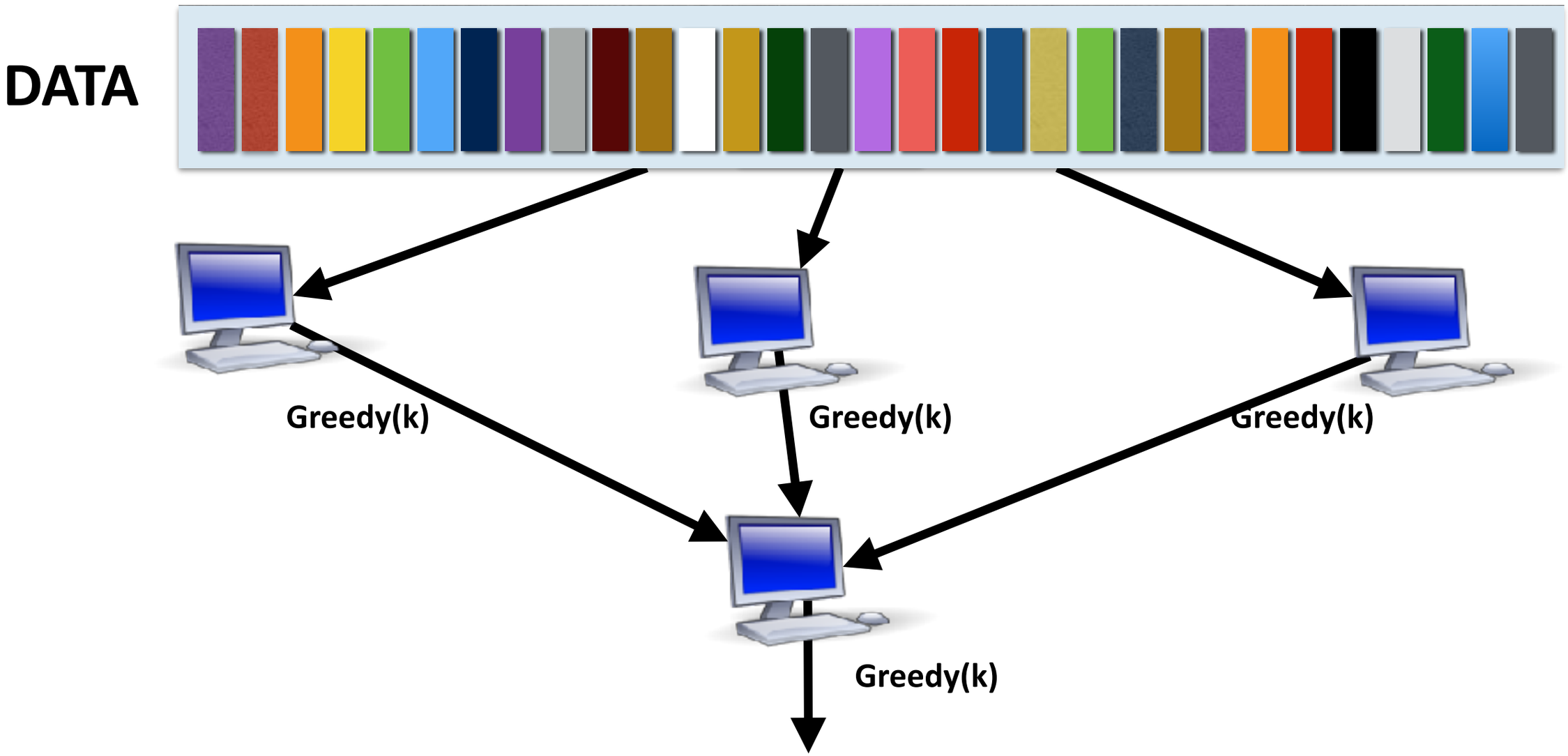
- Distributed

"Fast Greedy Algorithms in MapReduce and Streaming", 2013, Kumar, Moseley, Vassilvitskii, Vattani

Two-Stage Distributed Algorithm (GreeDi)

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

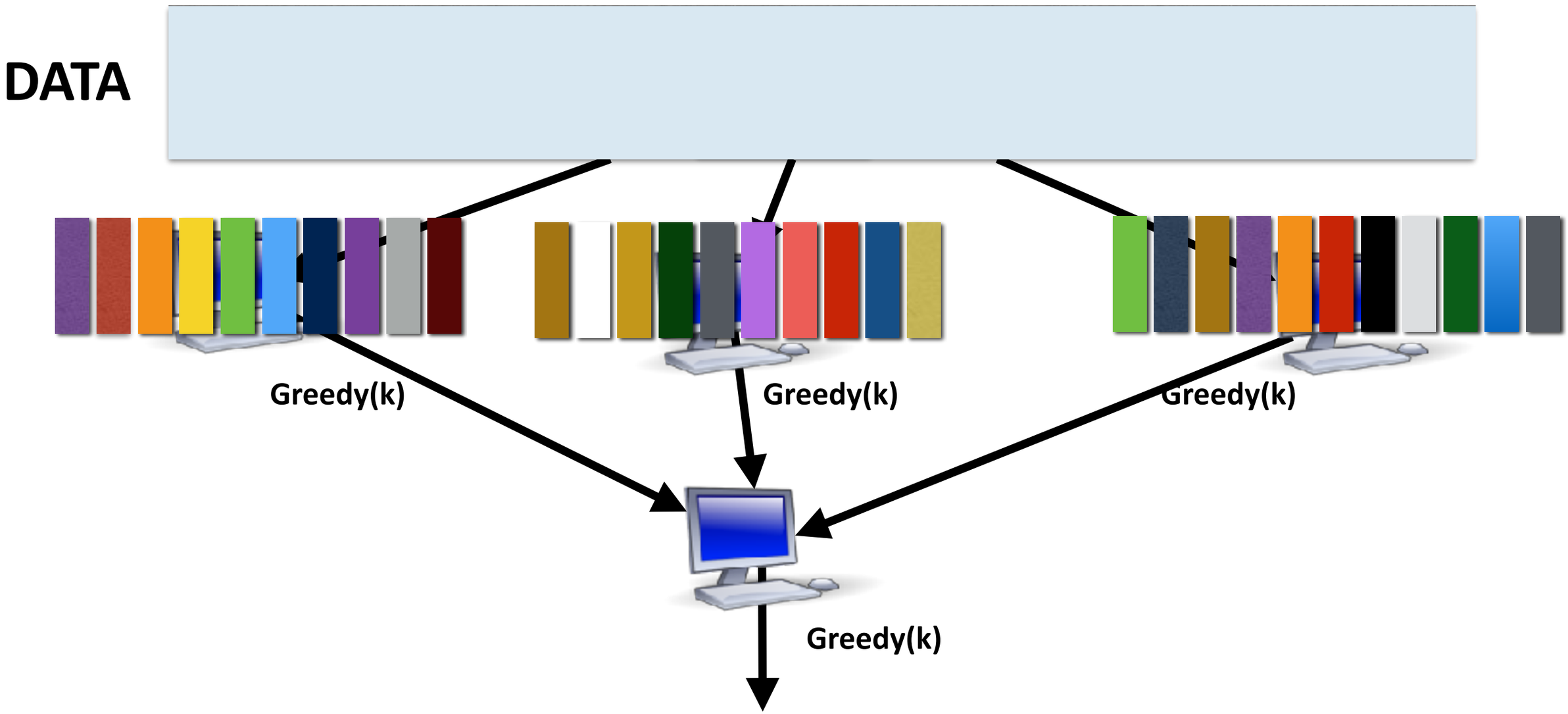


"Distributed Submodular Maximization: Identifying Representative Elements in Massive Data", Mirzasoleiman et al, 2013

Two-Stage Distributed Algorithm (GreeDi)

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

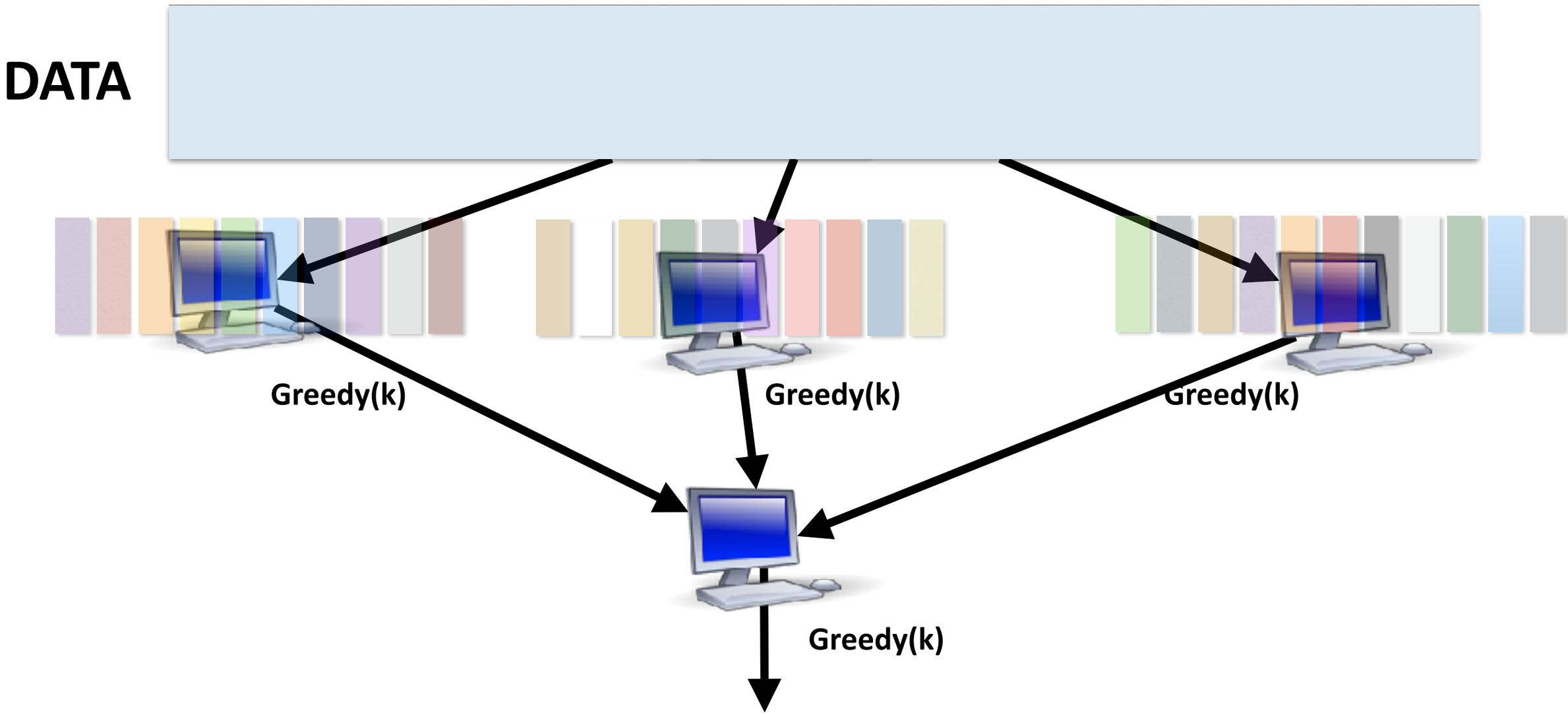


"Distributed Submodular Maximization: Identifying Representative Elements in Massive Data", Mirzasoleiman et al, 2013

Two-Stage Distributed Algorithm (GreeDi)

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

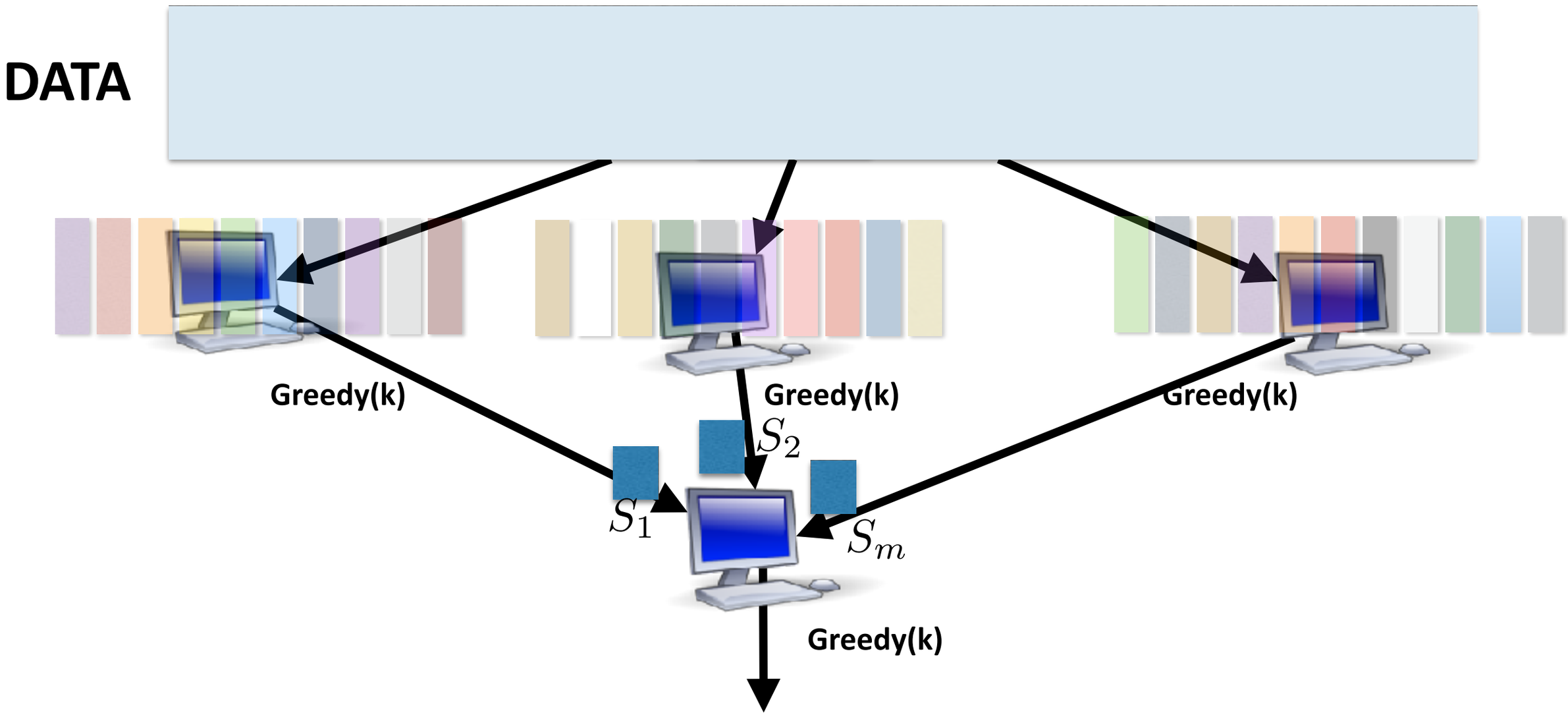


"Distributed Submodular Maximization: Identifying Representative Elements in Massive Data", Mirzasoleiman et al, 2013

Two-Stage Distributed Algorithm (GreeDi)

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

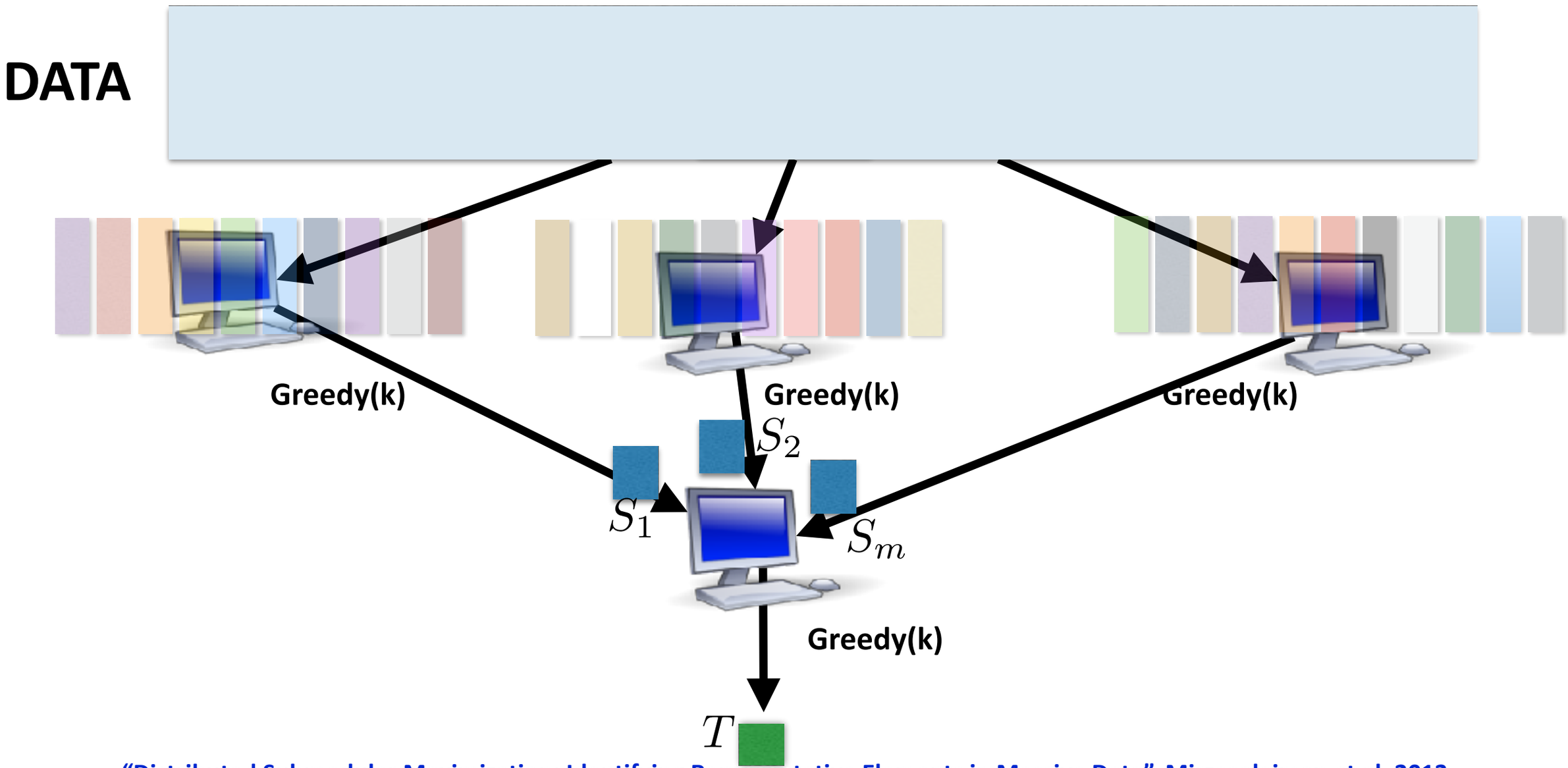


"Distributed Submodular Maximization: Identifying Representative Elements in Massive Data", Mirzasoleiman et al, 2013

Two-Stage Distributed Algorithm (GreeDi)

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$



"Distributed Submodular Maximization: Identifying Representative Elements in Massive Data", Mirzasoleiman et al, 2013

Theoretical Guarantees

- Split data (arbitrarily) among *m* machine

[Mirzasoleiman, Karbasi, Sarkar, Krause]

For monotone submodular functions, GreeDi gives

$$f(S_{\mathrm{GreeDi}}) \geq \frac{1-1/e}{\min(m,k)} \cdot \mathrm{OPT}$$

"Distributed Submodular Maximization: Identifying Representative Elements in Massive Data", 2013

"The Power of Randomization: Distributed Submodular Maximization on Massive Datasets", 2015.

Theoretical Guarantees

- Split data (arbitrarily) among *m* machine

[Mirzasoleiman, Karbasi, Sarkar, Krause]

For monotone submodular functions, GreeDi gives

$$f(S_{\mathrm{GreeDi}}) \geq \frac{1-1/e}{\min(m,k)} \cdot \mathrm{OPT}$$

"Distributed Submodular Maximization: Identifying Representative Elements in Massive Data", 2013

[Mirzasoleiman, Karbasi, Badanidiyuru, Krause] [Barbosa, Ene, Nguyen, Ward]

For monotone submodular functions, GreeDi gives

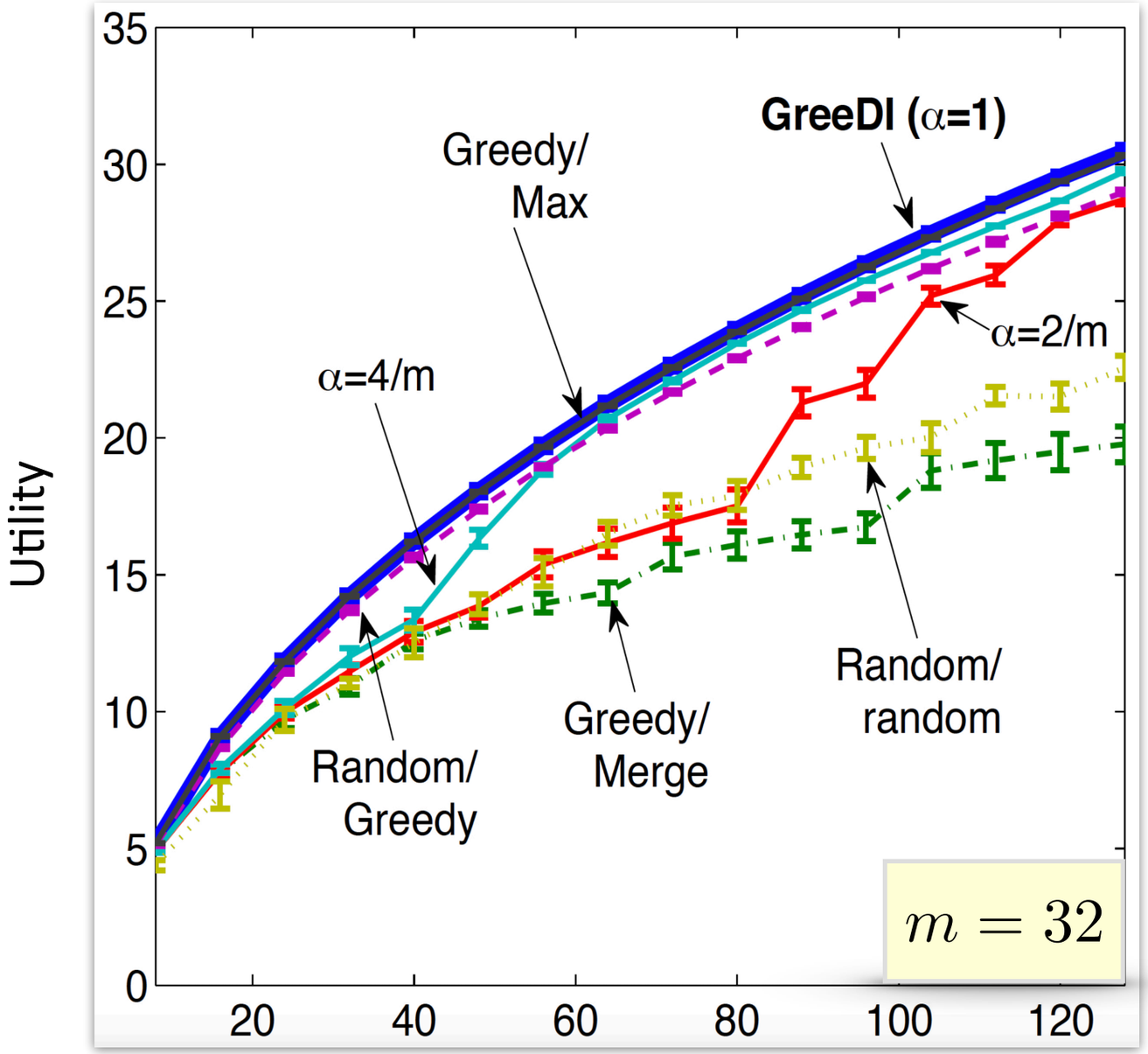
$$f(S_{\mathrm{GreeDi}}) \geq \frac{1-1/e}{\sqrt{\min(m,k)}} \cdot \mathrm{OPT}$$

and there exists no algorithm that can do better in general.

"Distributed Submodular Cover: Succinctly Summarizing Massive Data", 2015
"The Power of Randomization: Distributed Submodular Maximization on Massive Datasets", 2015.

Nonparametric Regression on Hadoop

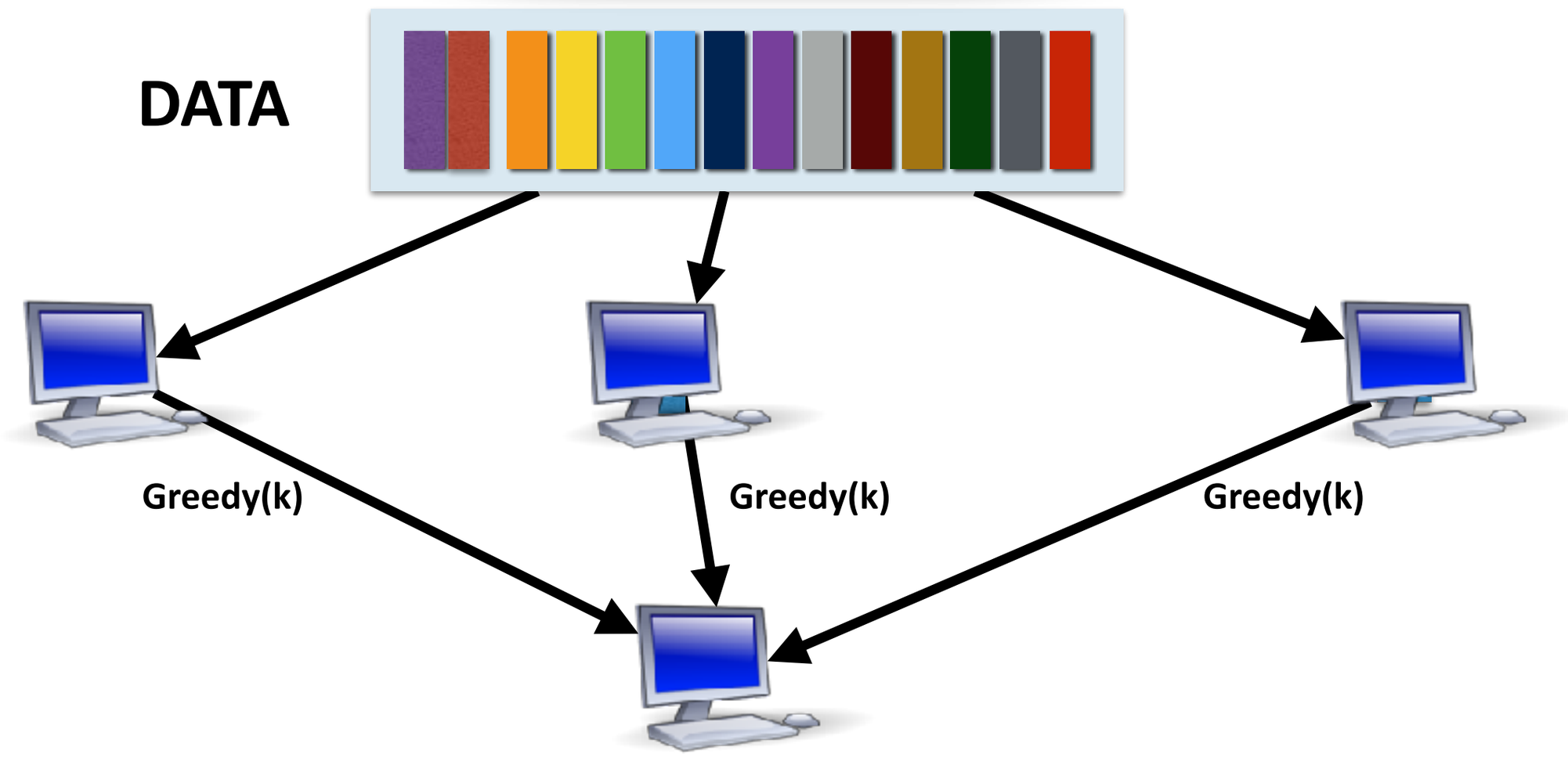
- **Yahoo! Webscope** dataset: 45,811,883 user click logs for news articles displayed in Yahoo! Front Page



The Power of Randomization

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

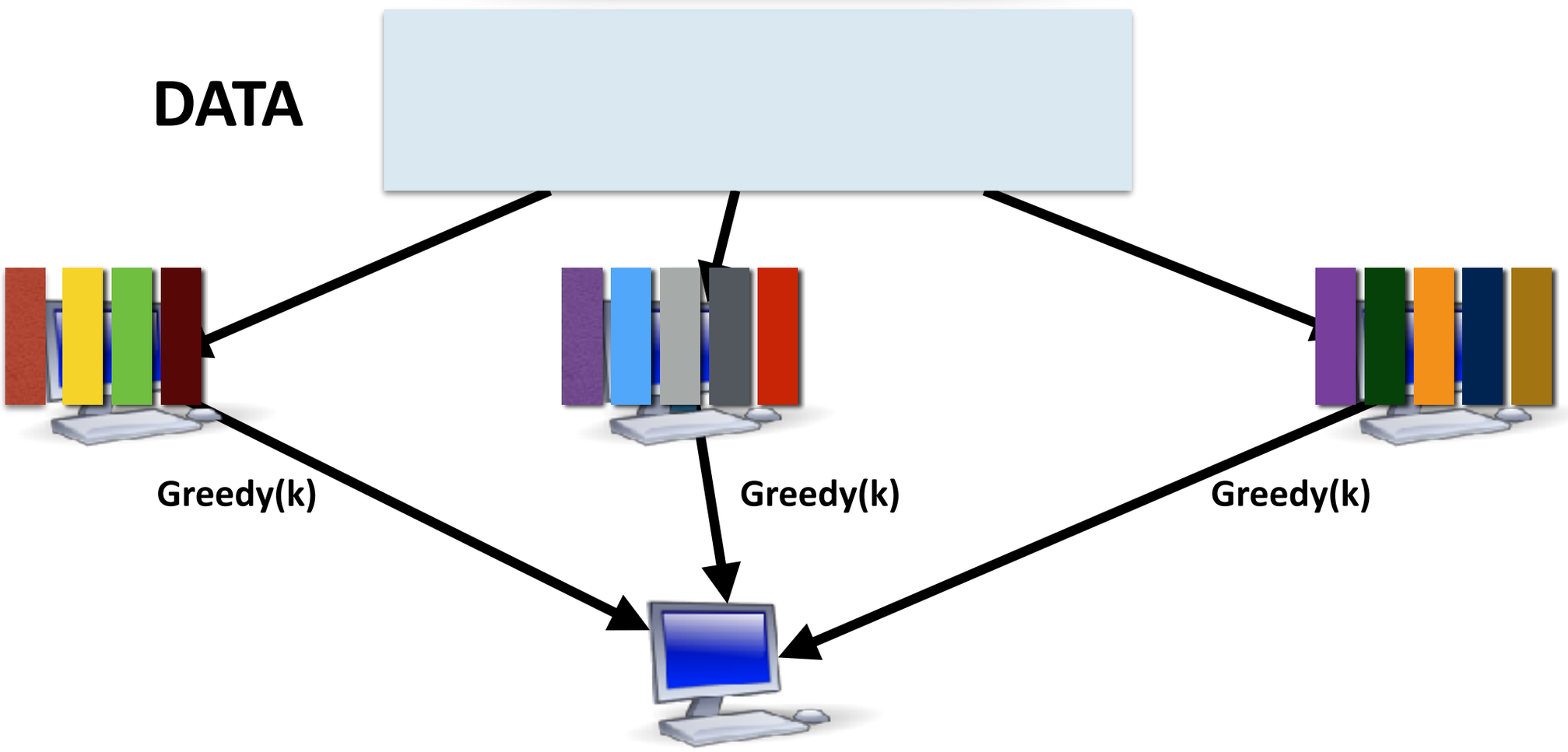


"The Power of Randomization: Distributed Submodular Maximization on Massive Datasets" Barbosa, Ene, Nguyen, Ward, 2015.
"Randomized composable core-sets for distributed submodular maximization" Mirrokni, Zadimoghaddam, 2015.

The Power of Randomization

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

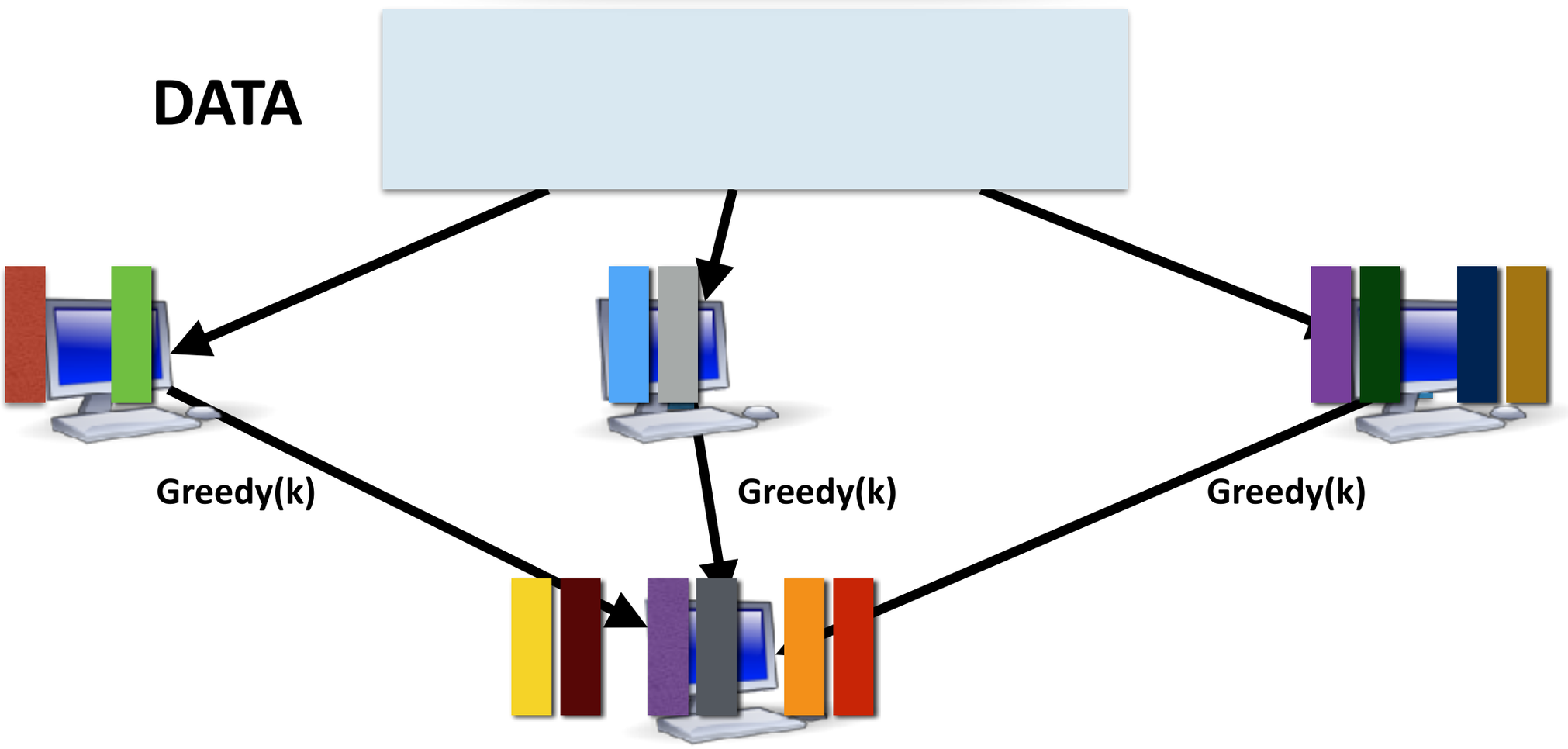


"The Power of Randomization: Distributed Submodular Maximization on Massive Datasets" Barbosa, Ene, Nguyen, Ward, 2015.
"Randomized composable core-sets for distributed submodular maximization" Mirrokni, Zadimoghaddam, 2015.

The Power of Randomization

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

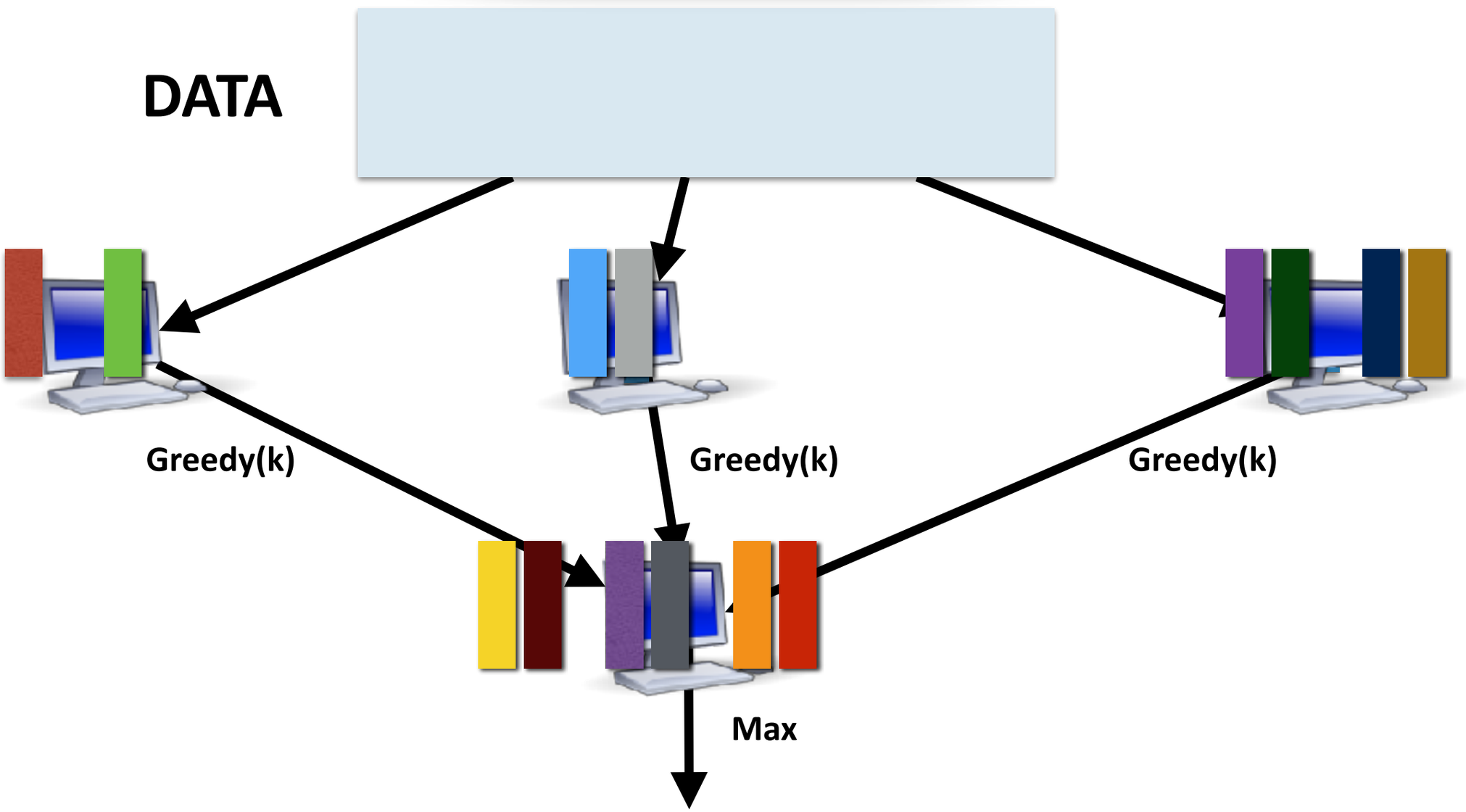


"The Power of Randomization: Distributed Submodular Maximization on Massive Datasets" Barbosa, Ene, Nguyen, Ward, 2015.
"Randomized composable core-sets for distributed submodular maximization" Mirrokni, Zadimoghaddam, 2015.

The Power of Randomization

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$



"The Power of Randomization: Distributed Submodular Maximization on Massive Datasets" Barbosa, Ene, Nguyen, Ward, 2015.
"Randomized composable core-sets for distributed submodular maximization" Mirrokni, Zadimoghaddam, 2015.

The Power of Randomization

- Consider the following problem where f is non-negative submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

[Barbosa, Ene, Nguyen, Ward] [Mirrokni, Zadimoghaddam]

When f is montone, radomized GreeDi returns a solution S of size k such that

$$\mathbb{E}[f(S_{\text{GreeDi}})] \geq \left(\frac{1-1/e}{2}\right) \text{OPT}$$

"The Power of Randomization: Distributed Submodular Maximization on Massive Datasets", 2015.

"Randomized composable core-sets for distributed submodular maximization" Mirrokni, Zadimoghaddam, 2015.

The Power of Randomization

- Consider the following problem where f is non-negative submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

[Barbosa, Ene, Nguyen, Ward] [Mirrokni, Zadimoghaddam]

When f is montone, radomized GreeDi returns a solution S of size k such that

$$\mathbb{E}[f(S_{\text{GreeDi}})] \geq \left(\frac{1-1/e}{2}\right) \text{OPT}$$

"The Power of Randomization: Distributed Submodular Maximization on Massive Datasets", 2015.

"Randomized composable core-sets for distributed submodular maximization" Mirrokni, Zadimoghaddam, 2015.

[Barbosa, Ene, Nguyen, Ward] [Mirrokni, Zadimoghaddam]

When f is non-montone, radomized GreeDi returns a solution S of size at most k such that

$$\mathbb{E}[f(S_{\text{GreeDi}})] \geq \left(1-\frac{1}{m}\right)\frac{1}{e}\left(1-\frac{1}{e}\right) \text{OPT}$$

"A New Framework for Distributed Submodular Maximization", 2016

Distributed Submodular Maximization

Arbitrary Partition

Distributed Submodular Maximization

Arbitrary Partition

Distributed Submodular Maximization: Identifying Representative Elements in Massive Data

Baharan Mirzasoleiman ETH Zurich
Amin Karbasi ETH Zurich
Rik Sarkar University of Edinburgh
Andreas Krause ETH Zurich

2013

Distributed Submodular Cover: Succinctly Summarizing Massive Data

Baharan Mirzasoleiman ETH Zurich
Amin Karbasi Yale University
Ashwinkumar Badanidiyuru Google
Andreas Krause ETH Zurich

2015

Distributed Submodular Maximization

Arbitrary Partition

Random Partition (2 Rounds)

Distributed Submodular Maximization: Identifying Representative Elements in Massive Data

Baharan Mirzasoleiman ETH Zurich
Amin Karbasi ETH Zurich
Rik Sarkar University of Edinburgh
Andreas Krause ETH Zurich

2013

The Power of Randomization
Distributed Submodular Maximization on Massive Datasets*

Rafael da Ponte Barbosa[1], Alina Ene[1], Huy L. Nguyễn[2], and Justin Ward[1,†]

2015

Distributed Submodular Cover: Succinctly Summarizing Massive Data

Baharan Mirzasoleiman ETH Zurich
Amin Karbasi Yale University
Ashwinkumar Badanidiyuru Google
Andreas Krause ETH Zurich

2015

Distributed Submodular Maximization

Arbitrary Partition

Random Partition (2 Rounds)

Distributed Submodular Maximization: Identifying Representative Elements in Massive Data

Baharan Mirzasoleiman ETH Zurich

Amin Karbasi ETH Zurich

Rik Sarkar University of Edinburgh

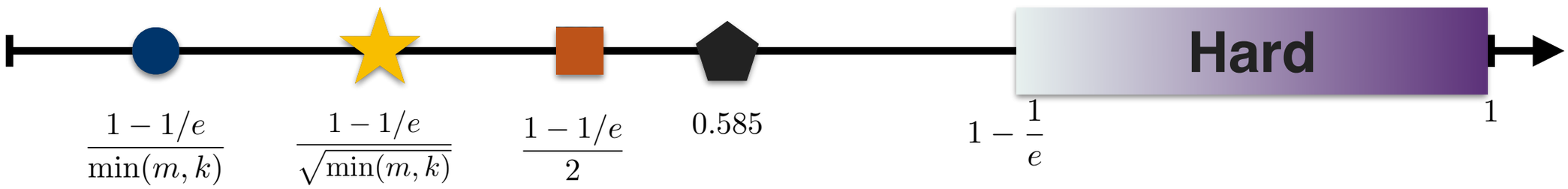
Andreas Krause ETH Zurich

2013

2015

The Power of Randomization
Distributed Submodular Maximization on Massive Datasets*

Rafael da Ponte Barbosa[1], Alina Ene[1], Huy L. Nguyễn[2], and Justin Ward[1,†]



Distributed Submodular Cover: Succinctly Summarizing Massive Data

2015

Baharan Mirzasoleiman ETH Zurich

Amin Karbasi Yale University

Ashwinkumar Badanidiyuru Google

Andreas Krause ETH Zurich

2015

Randomized Composable Core-sets for Distributed Submodular Maximization

Vahab Mirrokni Morteza Zadimoghaddam

Distributed Submodular Maximization

Arbitrary Partition

Random Partition (2 Rounds)

Distributed Submodular Maximization: Identifying Representative Elements in Massive Data

Baharan Mirzasoleiman ETH Zurich

Amin Karbasi ETH Zurich

Rik Sarkar University of Edinburgh

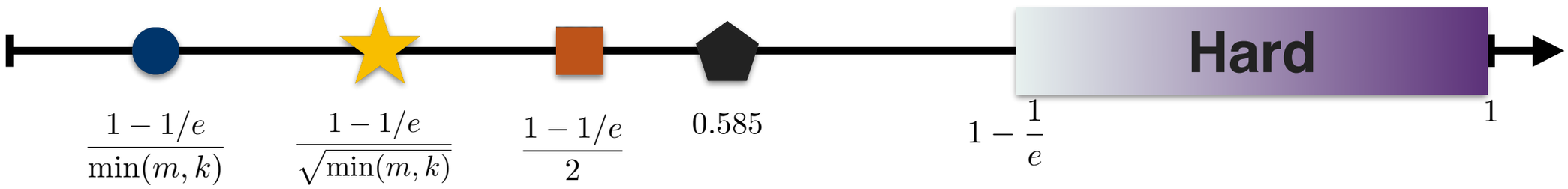
Andreas Krause ETH Zurich

2013

2015

The Power of Randomization
Distributed Submodular Maximization on Massive Datasets*

Rafael da Ponte Barbosa[1], Alina Ene[1], Huy L. Nguyễn[2], and Justin Ward[1,†]



Distributed Submodular Cover: Succinctly Summarizing Massive Data

Baharan Mirzasoleiman ETH Zurich

Amin Karbasi Yale University

Ashwinkumar Badanidiyuru Google

Andreas Krause ETH Zurich

2015

Randomized Composable Core-sets for Distributed Submodular Maximization

Vahab Mirrokni Morteza Zadimoghaddam

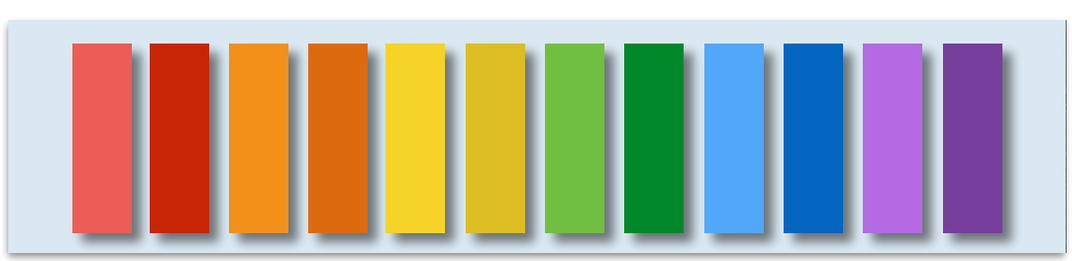
- Is it possible to achieve 1-1/e?

Multi-Stage Distributed Greedy

- Randomly distribute the data amongst m outer machines. Each machine then runs a greedy algorithm on the subset of data assigned to it.
- The central machine aggregates all these sub-solutions into one large solution.
- Randomly re-distribute the data amongst the m outer machines. Each machine then runs another greedy algorithm on the subset of data assigned to it *plus* the combined solution from the previous step.

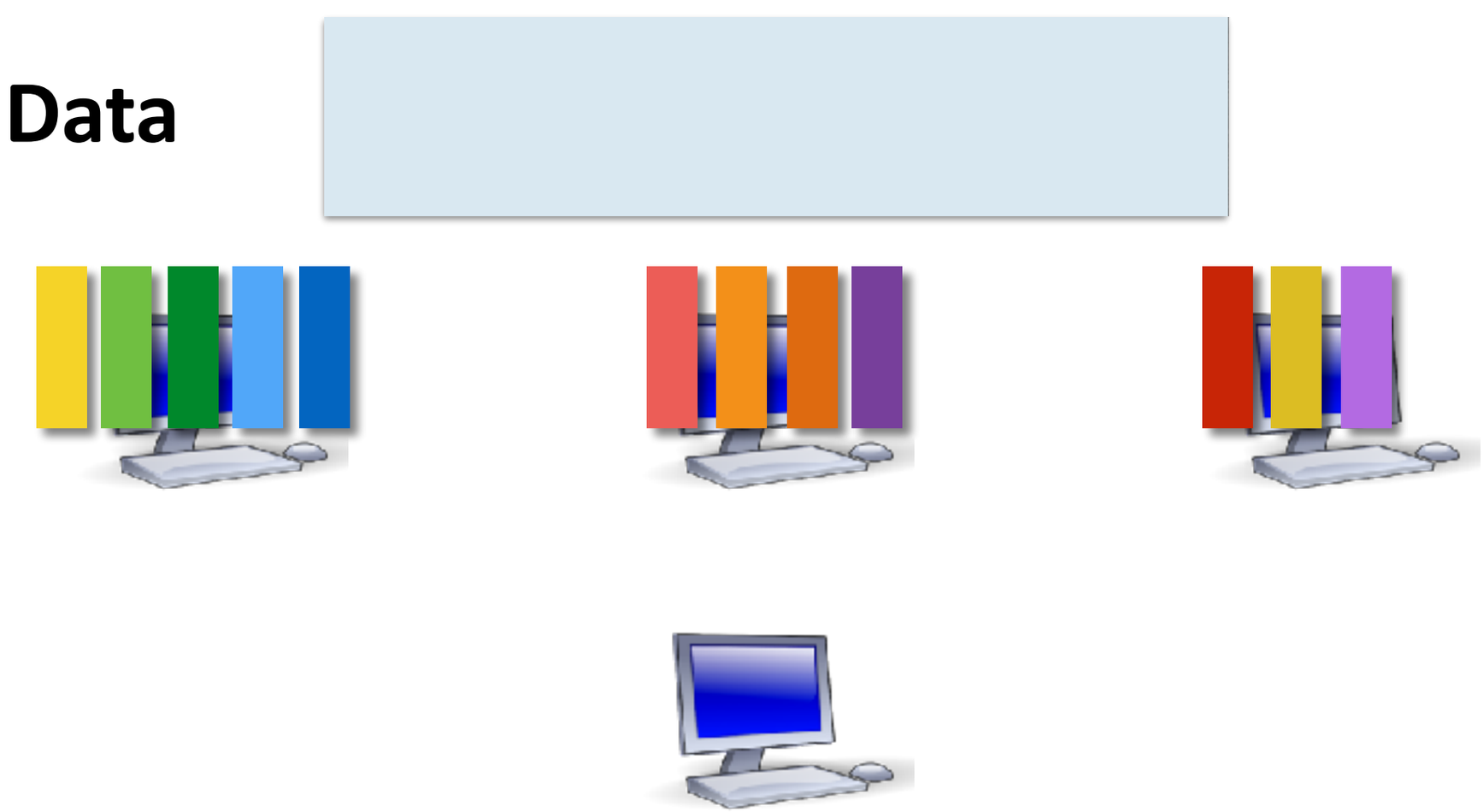
Multi-Stage Distributed Greedy

- Randomly distribute the data amongst m outer machines. Each machine then runs a greedy algorithm on the subset of data assigned to it.
- The central machine aggregates all these sub-solutions into one large solution.
- Randomly re-distribute the data amongst the m outer machines. Each machine then runs another greedy algorithm on the subset of data assigned to it *plus* the combined solution from the previous step.



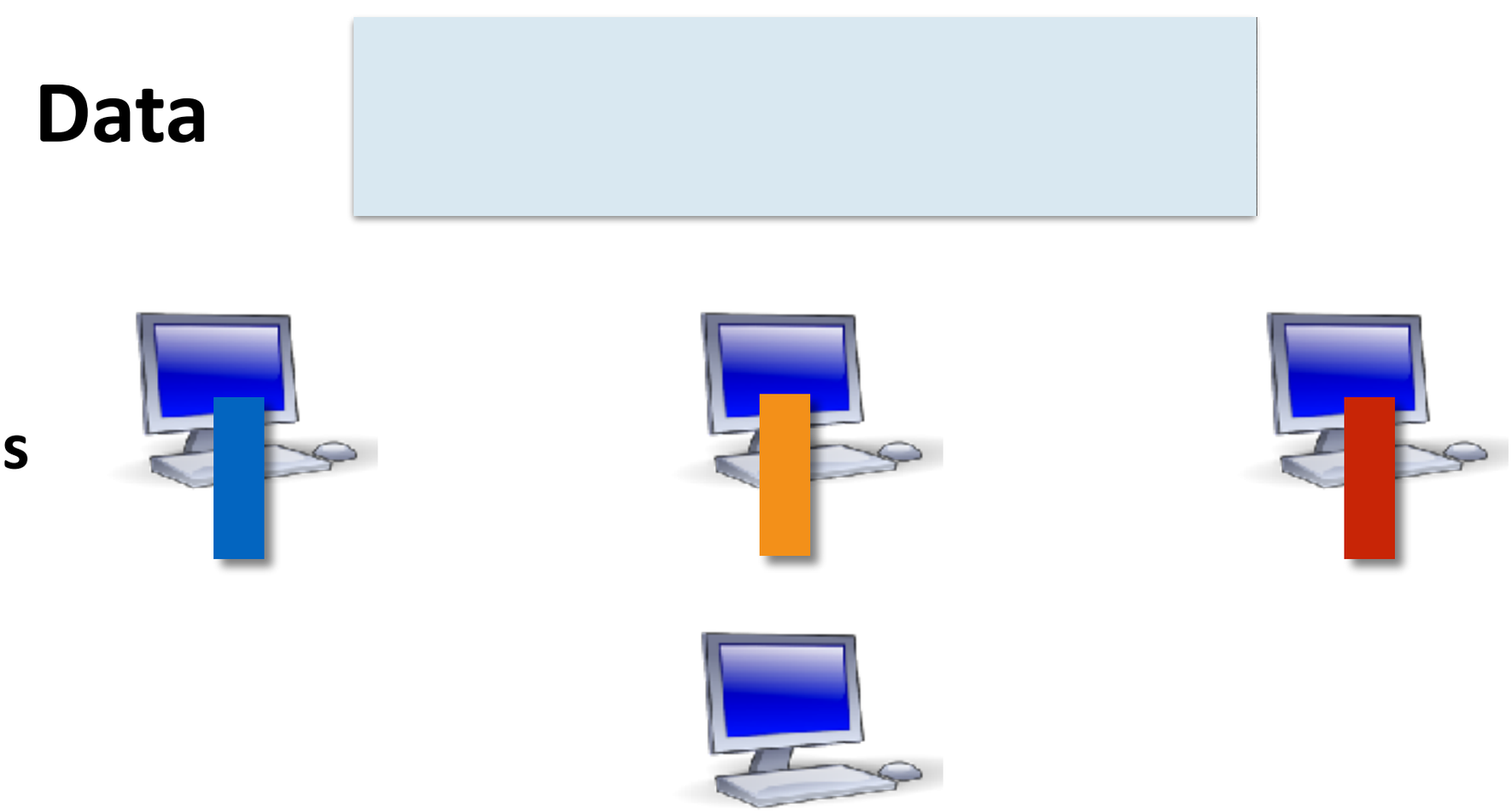
Multi-Stage Distributed Greedy

- Randomly distribute the data amongst m outer machines. Each machine then runs a greedy algorithm on the subset of data assigned to it.
- The central machine aggregates all these sub-solutions into one large solution.
- Randomly re-distribute the data amongst the m outer machines. Each machine then runs another greedy algorithm on the subset of data assigned to it *plus* the combined solution from the previous step.



Multi-Stage Distributed Greedy

- Randomly distribute the data amongst m outer machines. Each machine then runs a greedy algorithm on the subset of data assigned to it.
- The central machine aggregates all these sub-solutions into one large solution.
- Randomly re-distribute the data amongst the m outer machines. Each machine then runs another greedy algorithm on the subset of data assigned to it *plus* the combined solution from the previous step.

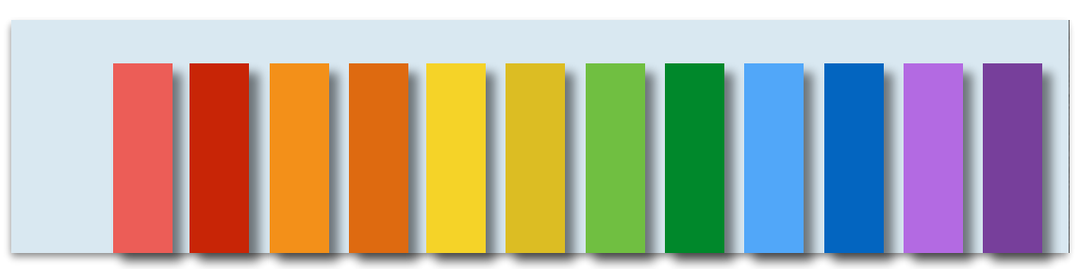


Multi-Stage Distributed Greedy

- Randomly distribute the data amongst m outer machines. Each machine then runs a greedy algorithm on the subset of data assigned to it.
- The central machine aggregates all these sub-solutions into one large solution.
- Randomly re-distribute the data amongst the m outer machines. Each machine then runs another greedy algorithm on the subset of data assigned to it *plus* the combined solution from the previous step.

Multi-Stage Distributed Greedy

- Randomly distribute the data amongst m outer machines. Each machine then runs a greedy algorithm on the subset of data assigned to it.
- The central machine aggregates all these sub-solutions into one large solution.
- Randomly re-distribute the data amongst the m outer machines. Each machine then runs another greedy algorithm on the subset of data assigned to it *plus* the combined solution from the previous step.

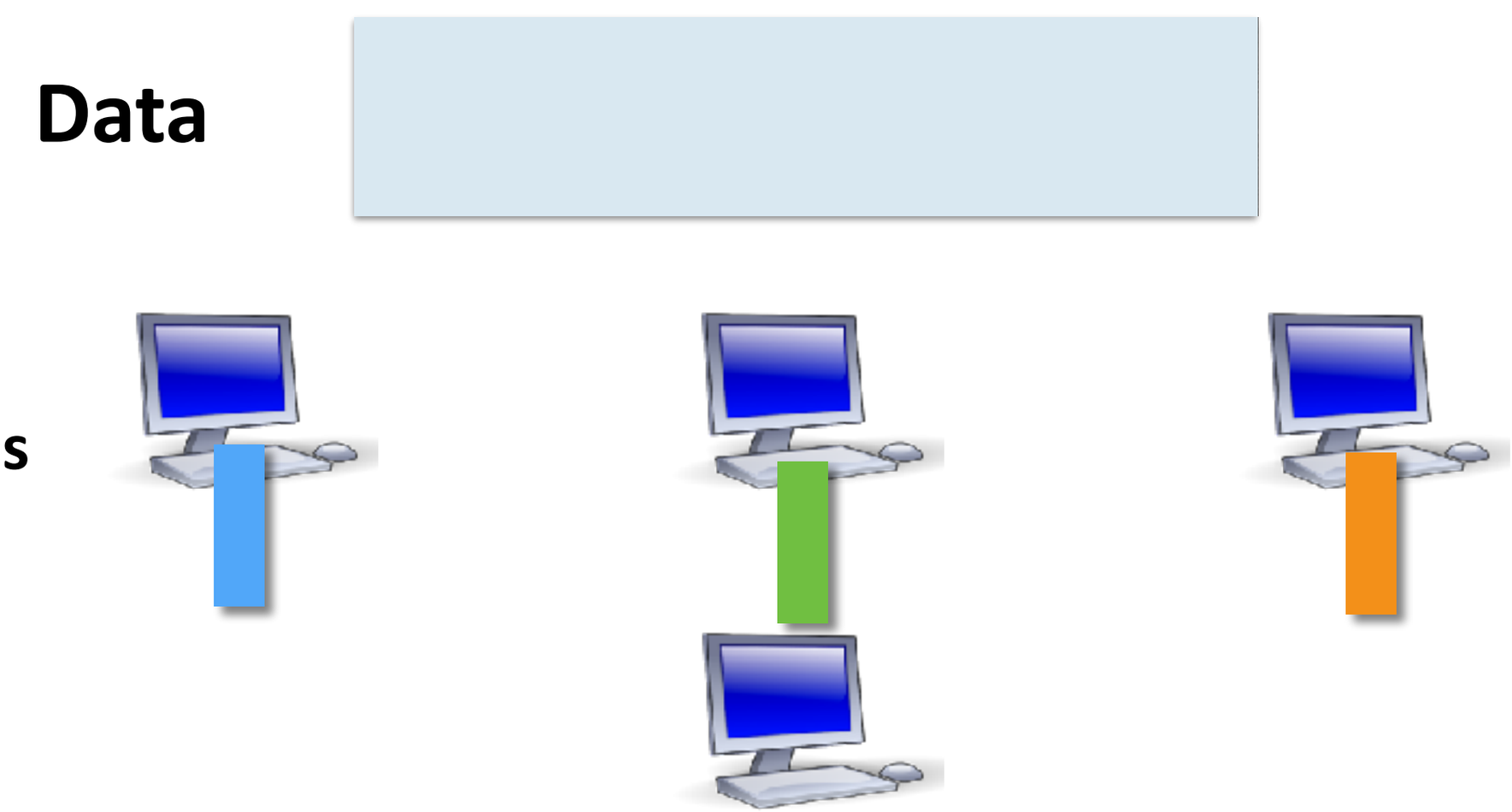


Multi-Stage Distributed Greedy

- Randomly distribute the data amongst m outer machines. Each machine then runs a greedy algorithm on the subset of data assigned to it.
- The central machine aggregates all these sub-solutions into one large solution.
- Randomly re-distribute the data amongst the m outer machines. Each machine then runs another greedy algorithm on the subset of data assigned to it *plus* the combined solution from the previous step.

Multi-Stage Distributed Greedy

- Randomly distribute the data amongst m outer machines. Each machine then runs a greedy algorithm on the subset of data assigned to it.
- The central machine aggregates all these sub-solutions into one large solution.
- Randomly re-distribute the data amongst the m outer machines. Each machine then runs another greedy algorithm on the subset of data assigned to it *plus* the combined solution from the previous step.



Distributed Submodular Maximization

Arbitrary Partition

Distributed Submodular Maximization: Identifying Representative Elements in Massive Data

Baharan Mirzasoleiman
ETH Zurich

Amin Karbasi
ETH Zurich

Rik Sarkar
University of Edinburgh

Andreas Krause
ETH Zurich

2013

Distributed Submodular Cover: Succinctly Summarizing Massive Data

Baharan Mirzasoleiman
ETH Zurich

Amin Karbasi
Yale University

Ashwinkumar Badanidiyuru
Google

Andreas Krause
ETH Zurich

2015

Distributed Submodular Maximization

Arbitrary Partition → Random Partition (2 Rounds) → Multi-Stage Algs

Distributed Submodular Maximization: Identifying Representative Elements in Massive Data

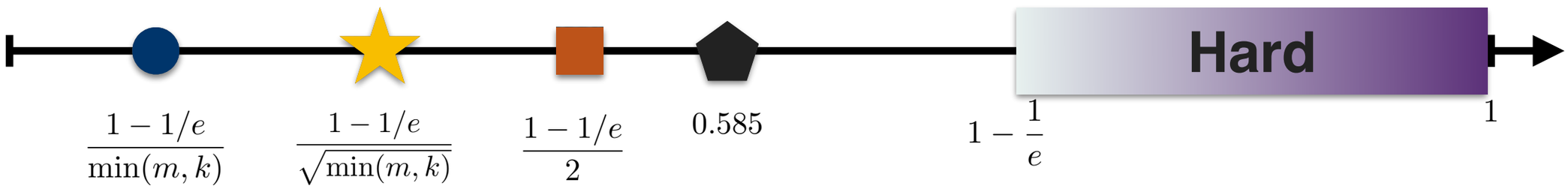
Baharan Mirzasoleiman ETH Zurich
Amin Karbasi ETH Zurich
Rik Sarkar University of Edinburgh
Andreas Krause ETH Zurich

2013

The Power of Randomization
Distributed Submodular Maximization on Massive Datasets*

Rafael da Ponte Barbosa[1], Alina Ene[1], Huy L. Nguyễn[2], and Justin Ward[1,†]

2015



Distributed Submodular Cover: Succinctly Summarizing Massive Data

Baharan Mirzasoleiman ETH Zurich
Amin Karbasi Yale University
Ashwinkumar Badanidiyuru Google
Andreas Krause ETH Zurich

2015

Randomized Composable Core-sets for Distributed Submodular Maximization

Vahab Mirrokni Morteza Zadimoghaddam

2015

Distributed Submodular Maximization

Arbitrary Partition → Random Partition (2 Rounds) → Multi-Stage Algs

Distributed Submodular Maximization: Identifying Representative Elements in Massive Data

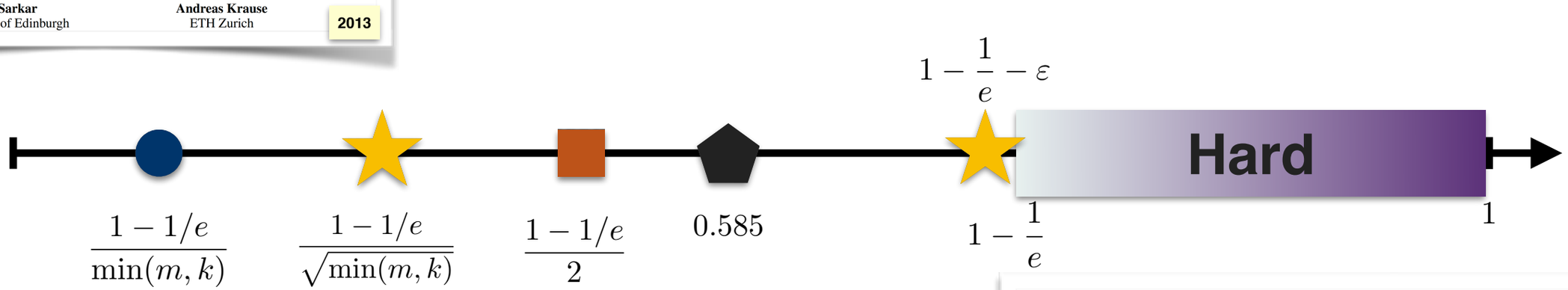
Baharan Mirzasoleiman ETH Zurich
Amin Karbasi ETH Zurich
Rik Sarkar University of Edinburgh
Andreas Krause ETH Zurich

2013

The Power of Randomization
Distributed Submodular Maximization on Massive Datasets*

Rafael da Ponte Barbosa[1], Alina Ene[1], Huy L. Nguyễn[2], and Justin Ward[1,†]

2015



Distributed Submodular Cover: Succinctly Summarizing Massive Data

Baharan Mirzasoleiman ETH Zurich
Amin Karbasi Yale University
Ashwinkumar Badanidiyuru Google
Andreas Krause ETH Zurich

2015

Randomized Composable Core-sets for Distributed Submodular Maximization

Vahab Mirrokni Morteza Zadimoghaddam

2015

A New Framework for Distributed Submodular Maximization

Rafael da Ponte Barbosa* Alina Ene† Huy L. Nguyễn‡ Justin Ward§

2017

$O(\frac{1}{\varepsilon})$ Rounds (with duplications)

Distributed Submodular Maximization

Arbitrary Partition → Random Partition (2 Rounds) → Multi-Stage Algs

Distributed Submodular Maximization: Identifying Representative Elements in Massive Data

Baharan Mirzasoleiman, ETH Zurich
Amin Karbasi, ETH Zurich
Rik Sarkar, University of Edinburgh
Andreas Krause, ETH Zurich

2013

The Power of Randomization
Distributed Submodular Maximization on Massive Datasets*

Rafael da Ponte Barbosa[1], Alina Ene[1], Huy L. Nguyễn[2], and Justin Ward[1,†]

2015

Regularized Submodular Maximization at Scale

Ehsan Kazemi* Shervin Minaee[†] Moran Feldman[‡] Amin Karbasi[§]

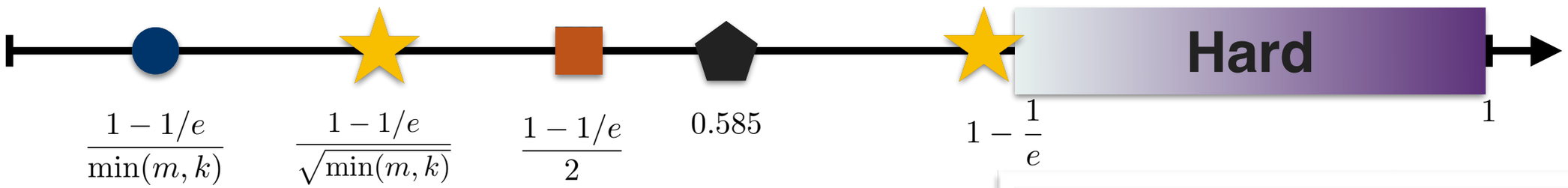
2020

Submodular Optimization in the MapReduce Model

Paul Liu and Jan Vondrak

2019

$1 - \frac{1}{e} - \varepsilon$ $O(\frac{1}{\varepsilon})$ Rounds (no duplications)



Distributed Submodular Cover: Succinctly Summarizing Massive Data

Baharan Mirzasoleiman, ETH Zurich
Amin Karbasi, Yale University
Ashwinkumar Badanidiyuru, Google
Andreas Krause, ETH Zurich

2015

Randomized Composable Core-sets for Distributed Submodular Maximization

Vahab Mirrokni Morteza Zadimoghaddam

2015

A New Framework for Distributed Submodular Maximization

Rafael da Ponte Barbosa* Alina Ene† Huy L. Nguyễn‡ Justin Ward§

2017

$O(\frac{1}{\varepsilon})$ Rounds (with duplications)

Multi-Stage Distributed Greedy: Open Question

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

[Liu, Vondrak] [Kazemi, Minaee, Feldman, Karbasi]

Multi-Stage GreeDi, after $O(1/\epsilon)$ iterations, and without duplicating the data returns a solution of size at most k such that

$$\mathbb{E}[f(S_{\text{Multi-GreeDi}})] \geq (1-\epsilon)\left(1-\frac{1}{e}\right)\text{OPT}$$

"Submodular Optimization in MapReduce Model", 2019

"Regularized Submodular Maximiza at Scale", 2020

Multi-Stage Distributed Greedy: Open Question

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

[Liu, Vondrak] [Kazemi, Minaee, Feldman, Karbasi]

Multi-Stage GreeDi, after $O(1/\epsilon)$ iterations, and without duplicating the data returns a solution of size at most k such that

$$\mathbb{E}[f(S_{\text{Multi-GreeDi}})] \geq (1-\epsilon)\left(1-\frac{1}{e}\right)\text{OPT}$$

"Submodular Optimization in MapReduce Model", 2019

"Regularized Submodular Maximiza at Scale", 2020

- Can we obtain (1-1/e) with fewer iteration?

Distributed Submodular Maximization: Extensions

- More Complex constraints

  "Distributed Submodular Maximization", Mirzasoleiman, Karbasi, Sarkar, Krause, 2016.

  "The Power of Randomization: Distributed Submodular Maximization on Massive Datasets" Barbosa, Ene, Nguyen, Ward, 2015.

- Non-monotone:

  "Parallel Double Greedy Submodular Maximization", Pan, Jegelka, Gonzalez, Bradley, Jordan, 2014.

  "The Power of Randomization: Distributed Submodular Maximization on Massive Datasets" Barbosa, Ene, Nguyen, Ward, 2015.

- Filteration:

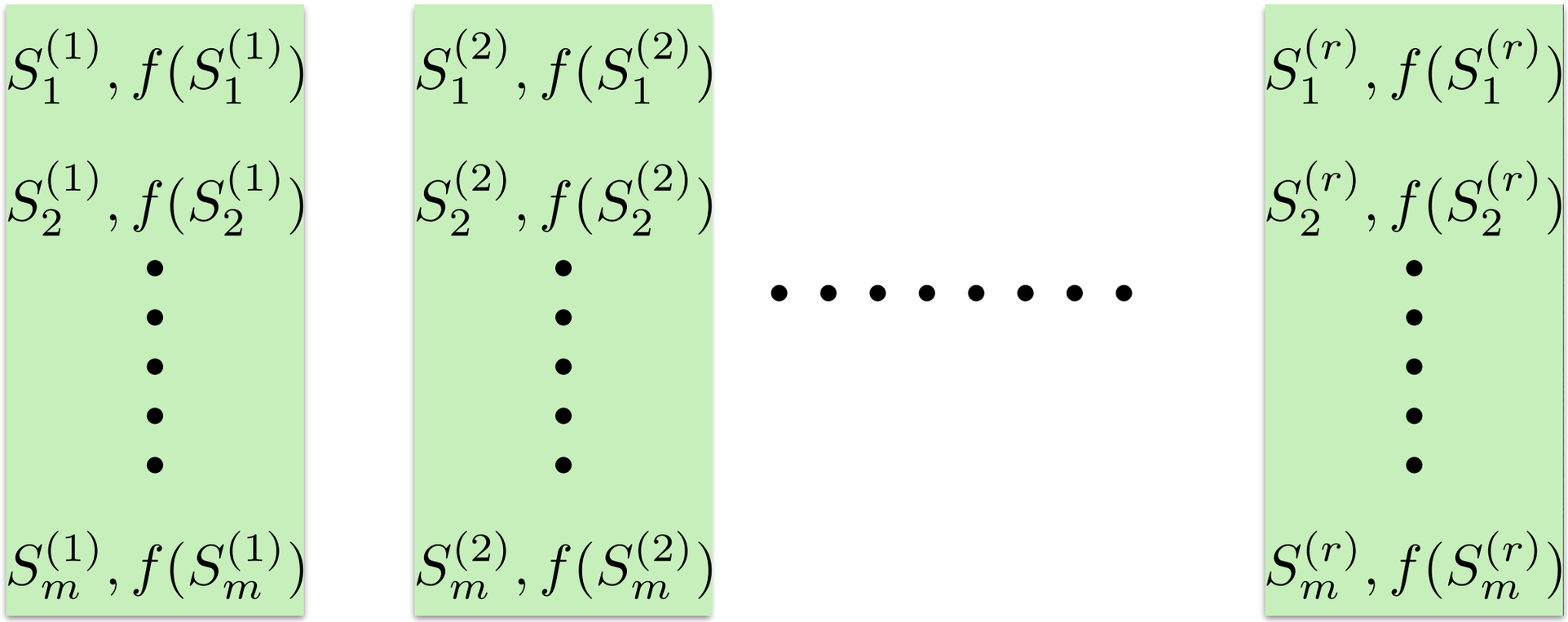
  "Fast Greedy Algorithms in MapReduce and Streaming", Kumar, Moseley, Vassilvitskii, Vattani, 2013.

- Sketching:

  "Optimal Distributed Submodular Optimization via Sketching", Bateni, Esfandiari, Mirrokni, 2018.

Adaptivity Complexity

- **r-adaptive**: An algorithm is r-adaptive if it makes r rounds of parallel computations. In each round, the algorithm is allowed to make polynomially many queries.

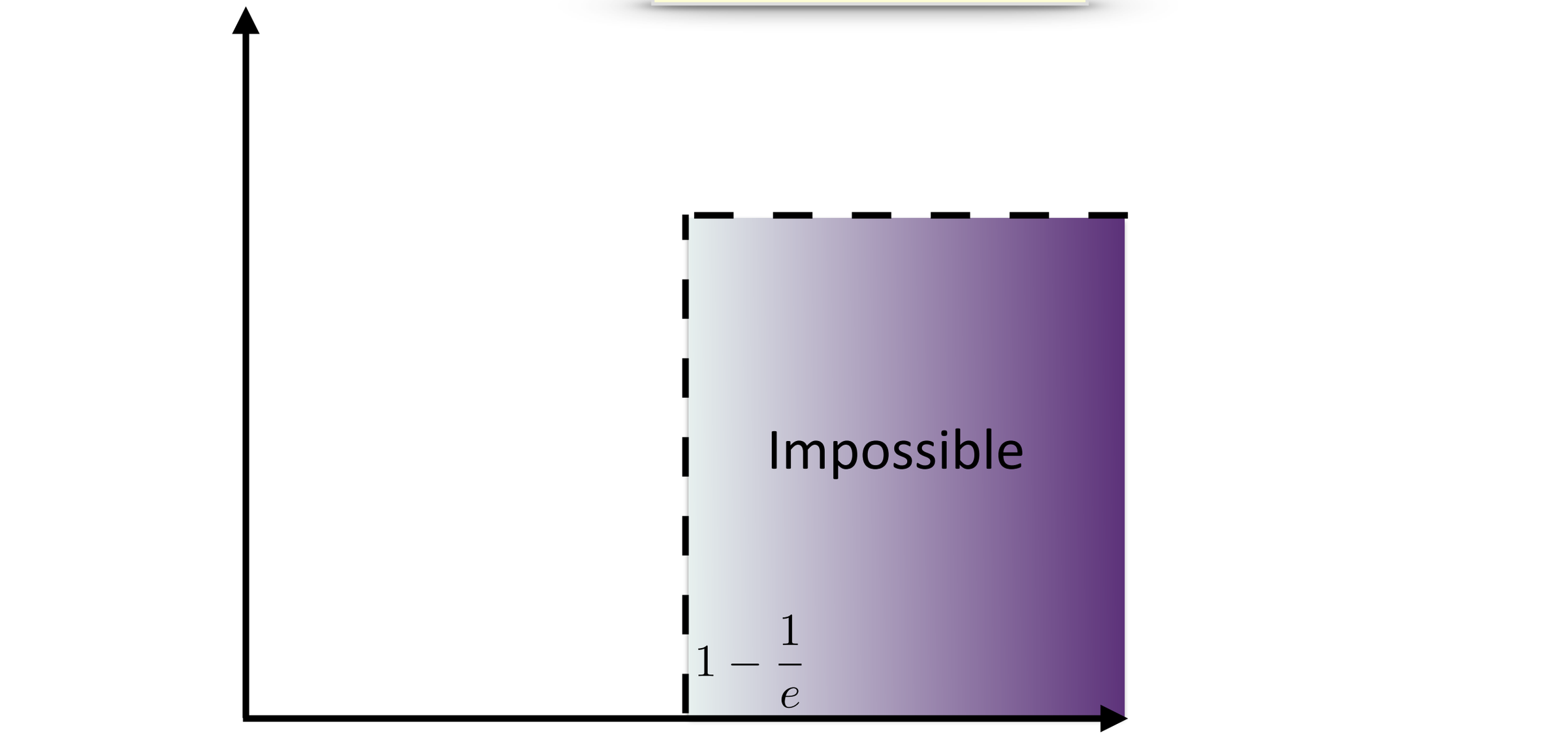


"The adaptive complexity of maximizing a submodular function", Balkanski, Singer, 2018

Adaptivity Complexity

- Consider the following problem where f is monotone submodular:

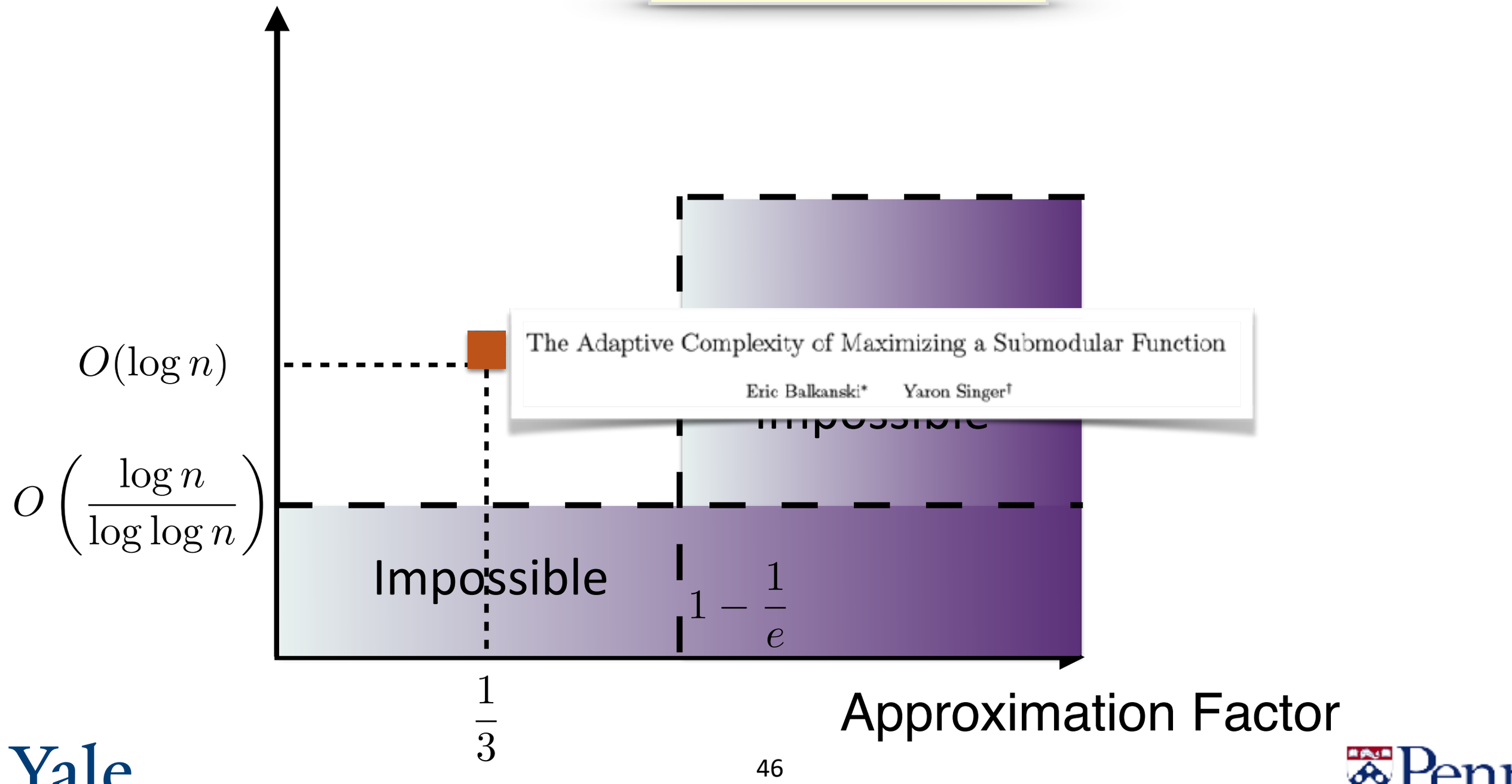
$$S^* = \arg\max_{|S| \leq k} f(S)$$



Adaptivity Complexity

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

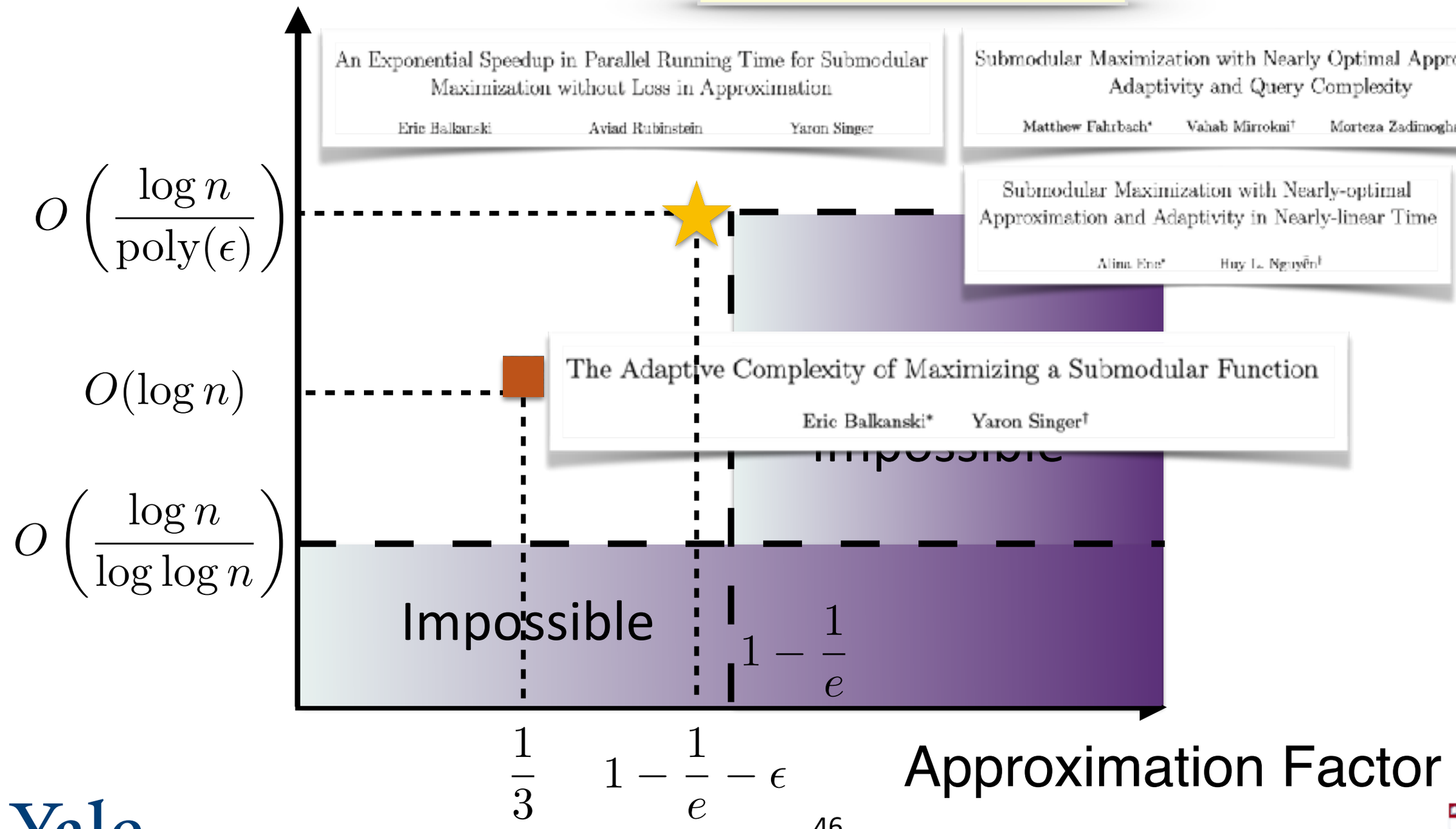


Adaptivity Complexity

- Consider the following problem where f is monotone submodular:

$$S^* = \arg\max_{|S| \leq k} f(S)$$

Adaptivity Complexity



Adaptivity Complexity: Extensions

Adaptivity Complexity: Extensions

- Matroid Constraints

  "An optimal approximation for submodular maximization under a matroid constraint in the adaptive complexity model", Balkanski, Rubinstein, Singer , 2019

  "An optimal approximation for submodular maximization under a matroid constraint in the adaptive complexity model", Balkanski, Rubinstein, Singer , Chekuri, Quanrud, 2019.

  "Submodular maximization with matroid and packing constraints in parallel", Ene, Nguyen, Vladu, 2019.

- Unconstrained Submodular Maximization

  "Unconstrained submodular maximization with constant adaptive complexity", Chen, Feldman, Karbasi, 2019.

- Non-montone Submodular Maximization

  "Non-monotone submodular maximization in exponentially fewer iterations", Balkanski, Breuer, Singer, 2018

  "Non-monotone submodular maximization with nearly optimal adaptivity and query complexity", Fahrbach, Mirrokni, Zadimoghaddam, 2019.

Adaptivity Complexity: Extensions

- Matroid Constraints

  "An optimal approximation for submodular maximization under a matroid constraint in the adaptive complexity model", Balkanski, Rubinstein, Singer , 2019

  "An optimal approximation for submodular maximization under a matroid constraint in the adaptive complexity model", Balkanski, Rubinstein, Singer , Chekuri, Quanrud, 2019.

  "Submodular maximization with matroid and packing constraints in parallel", Ene, Nguyen, Vladu, 2019.

- Unconstrained Submodular Maximization

  "Unconstrained submodular maximization with constant adaptive complexity", Chen, Feldman, Karbasi, 2019.

- Non-montone Submodular Maximization

  "Non-monotone submodular maximization in exponentially fewer iterations", Balkanski, Breuer, Singer, 2018

  "Non-monotone submodular maximization with nearly optimal adaptivity and query complexity", Fahrbach, Mirrokni, Zadimoghaddam, 2019.

- Convex Minimization

  "Parallelization does not accelerate convex optimization: Adaptivity lower bounds for non-smooth convex minimization" Balkanski, Singer, 2019

  "Lower Bounds for Parallel and Randomized Convex Optimization", Diakonikolas, Guzmán, 2019

  "Complexity of Highly Parallel Non-Smooth Convex Optimization", Bubeck , Jiang , Lee, Li, Sidford, 2019

- Interactive Stochastic Optimization

  "Adaptivity in Adaptive Submodularity", Hossein Esfandiari, Karbasi, Mirrokni, 2019

Where Is the Oracle?

There is no **oracle**

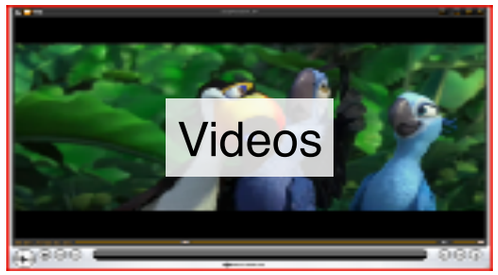
Stochastic Oracle

- Very often, we cannot evaluate the function *exactly*

Stochastic Oracle

- Very often, we cannot evaluate the function *exactly*
  - The value has to be estimated from the data (empirical mean)

$$f(S) = \frac{1}{n}\sum_{i=1}^{n} f_i(S)$$



Data Summarization:

Extract small, representative subset out of a massive data set

Yale

Penn

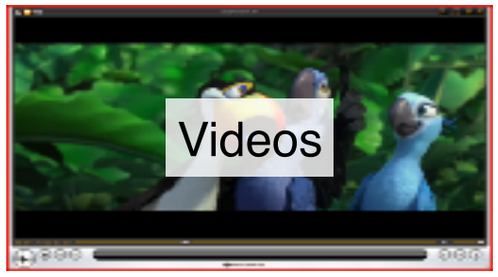
Stochastic Oracle

- Very often, we cannot evaluate the function *exactly*
  - The value has to be estimated from the data (empirical mean)

$$f(S) = \frac{1}{n}\sum_{i=1}^{n} f_i(S)$$

  - It is defined through a stochastic process

$$f(S) = \mathbb{E}_{\theta \sim D}[f_\theta(S)]$$



Data Summarization:

Extract small, representative subset out of a massive data set

Influence Maximization:

Find a small subset nodes with large influence

Part Two

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

Perfect Oracle

Discrete Methods

- **Centralized**
- **Streaming**
- **Distributed**

Imperfect Oracle

Continuous Methods