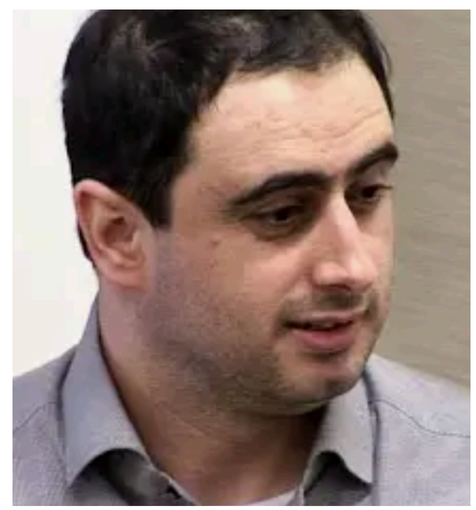
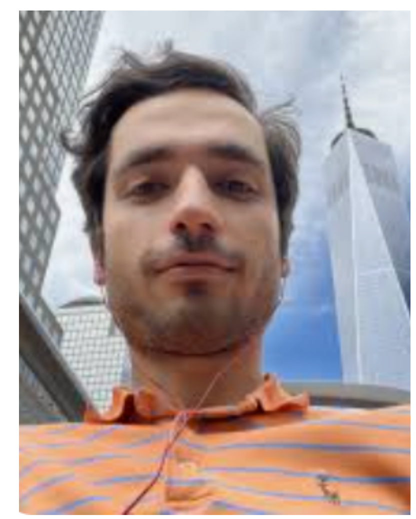
Technische Universiteit Eindhoven University of Technology

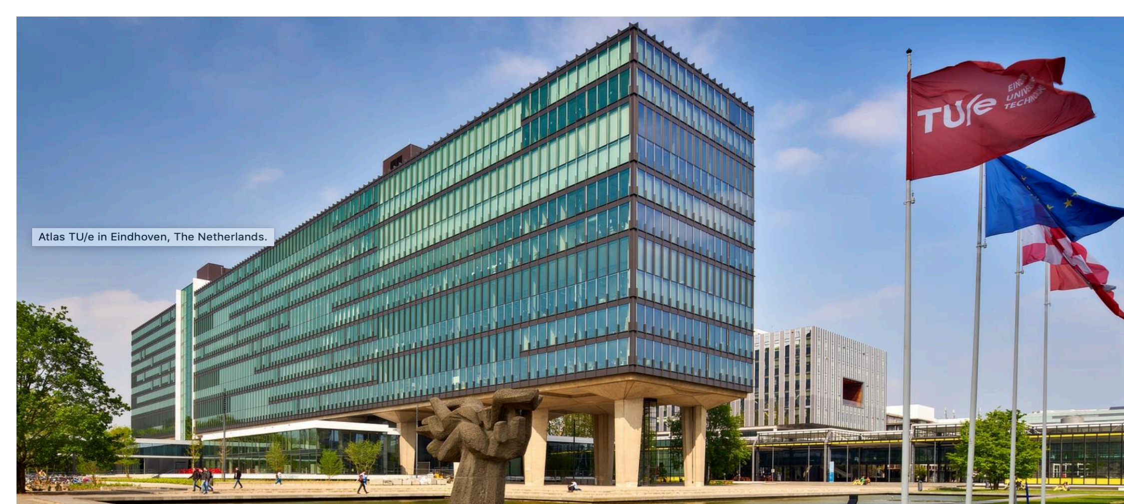
Where innovation starts

A Dynamic Algorithm for Weighted Submodular Cover Problem




Kiarash Banihashem   Samira Goudarzi   MohammadTaghi Hajiaghayi   Peyman Jabbarzade   Morteza Monemizadeh



Submodular cover

Input:

V : Ground set of elements

$f : 2^V \to \mathbb{R}^{\geq 0}$: a monotone submodular function

$w : V \to \mathbb{R}^{\geq 0}$: a weight function

Marginal gain

$$\Delta(v \mid A) := f(A \cup v) - f(A)$$

Submodular Function →

$$\Delta(v \mid A) \geq \Delta(v \mid B)$$

$\forall A \subseteq B \subseteq V$

Submodular cover

Input:

V : Ground set of elements

$f : 2^V \rightarrow \mathbb{R}^{\geq 0}$: a monotone submodular function

$w : V \rightarrow \mathbb{R}^{\geq 0}$: a weight function

Cost:

Compute a set $S \subseteq V$ minimizes the cost

$$cost(S) = \sum_{v \in S} w(s)$$

Constraint:

$$f(S) = f(V)$$

Density of an element

An important concept in our dynamic algorithm is the density of an element:

- Density of an element: $d(v) := \dfrac{f(v)}{w(v)}$ $\quad \forall v \in V$

- Marginal density of an element: $d(v\,|\,A) := \dfrac{\Delta(v\,|\,A)}{w(v)}$ $\quad \forall A \subseteq V, v \in V$

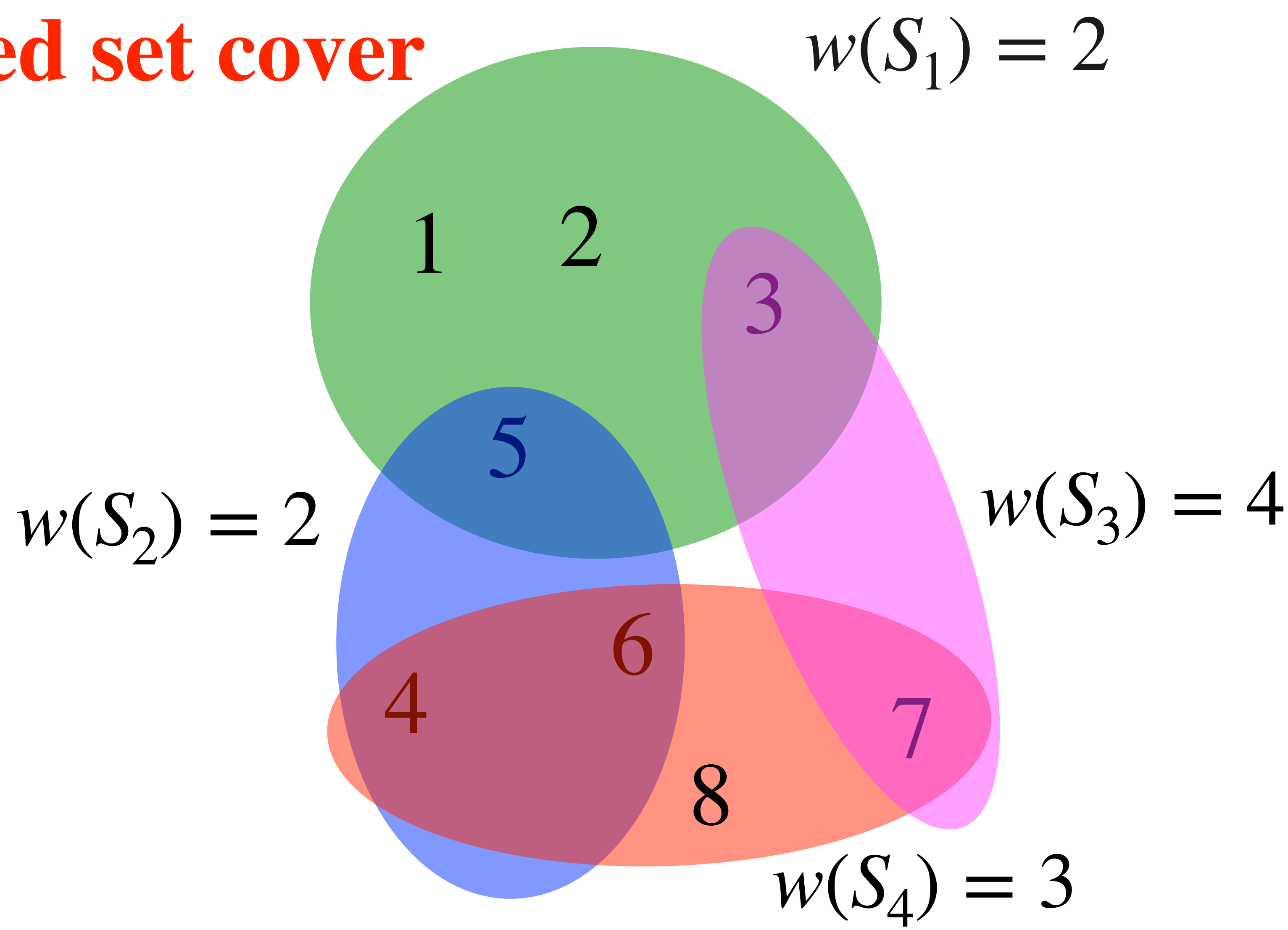
Example: Weighted set cover

$S_1 = \{1,2,3,5\}$

$S_2 = \{4,5,6\}$

$S_3 = \{3,7\}$

$S_4 = \{4,6,7,8\}$



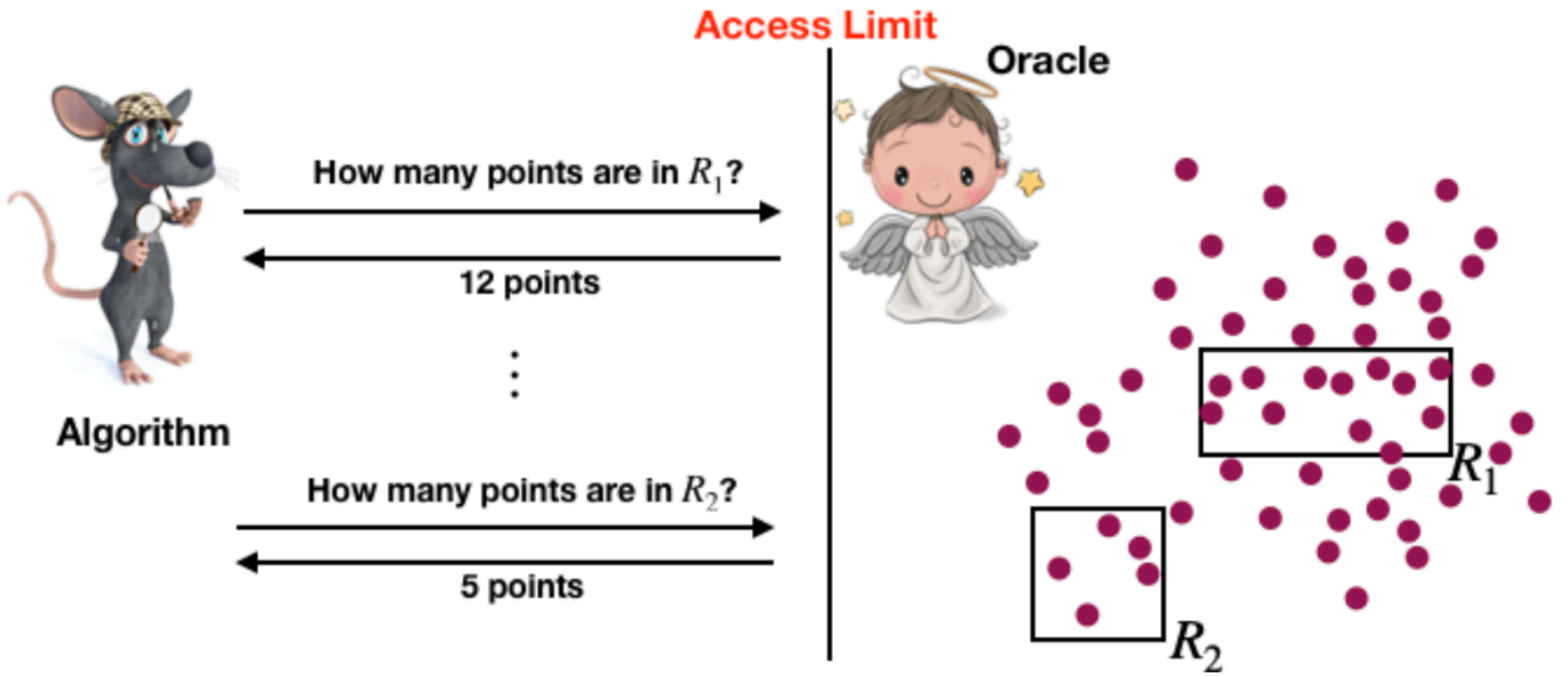
Ground set: $V = \{S_1, S_2, S_3, S_4\}$

Universe set: $U = \{1,2,3,4,5,6,7,8\}$

Cost: $cost(\mathcal{S}) = w(S_1) + w(S_4) = 5$

Constraint: $f(\mathcal{S}) = f(\{S_1, S_4\}) = f(V)$

Query access model



- The algorithm asks **queries** and an **oracle** **responds**.
- The **complexity** of the model is measured using the number of queries that the algorithm can make.

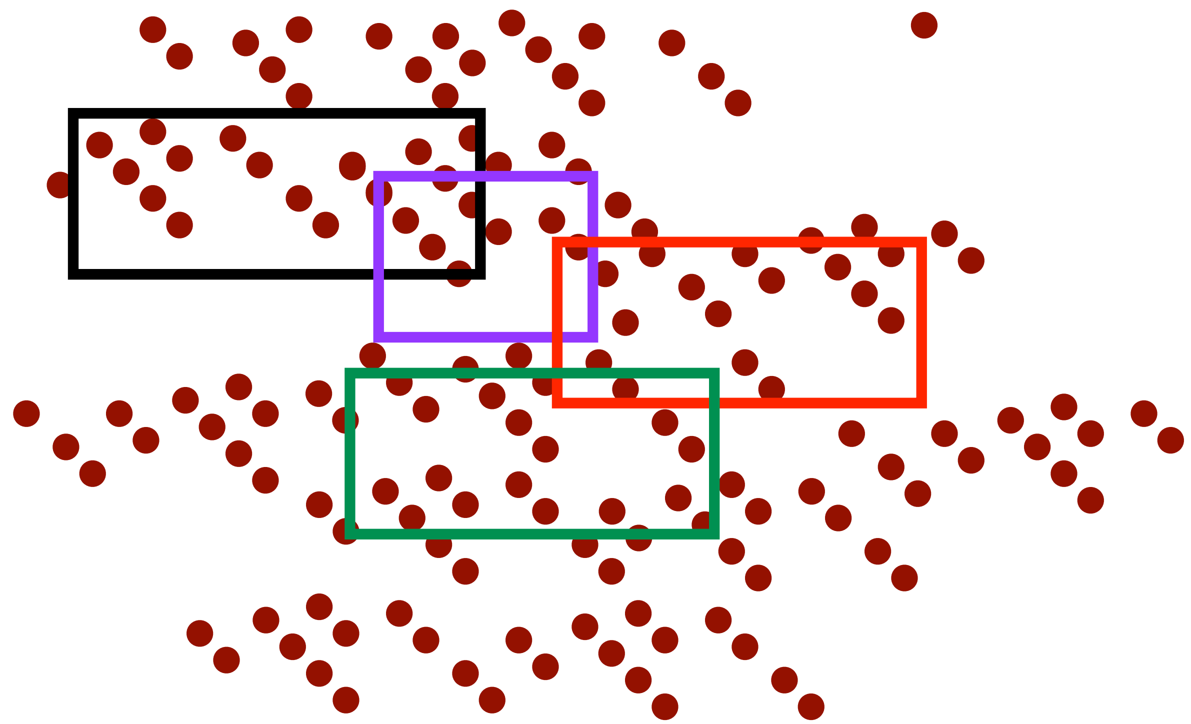
Bicriteria

A set $\mathcal{S}$ is called a $(1-\epsilon, c)$-bicriteria approximate solution if it satisfies

- $f(\mathcal{S}) \geq (1-\epsilon) \cdot f(V)$
- $cost(\mathcal{S}) \leq c \cdot cost(\mathcal{S}_{opt})$

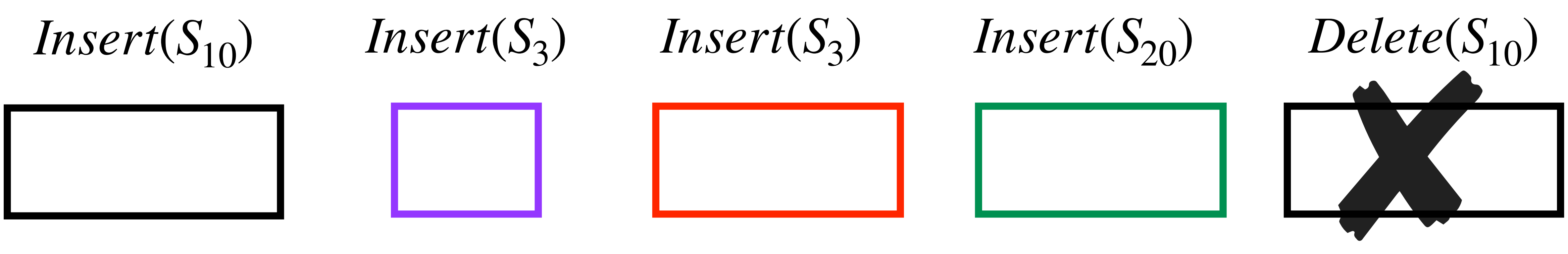
$\mathcal{S}_{opt}$: the optimal solution

Dynamic model

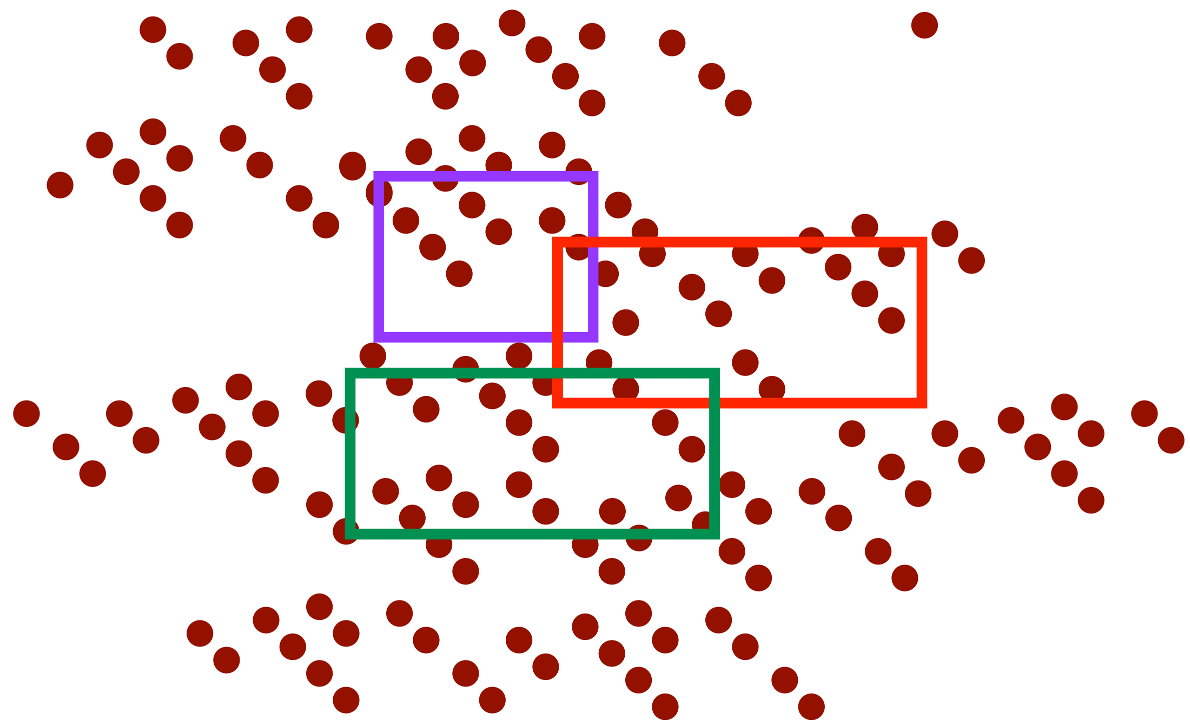


Solution:

S_{10}, S_3, S_{20}



Dynamic model

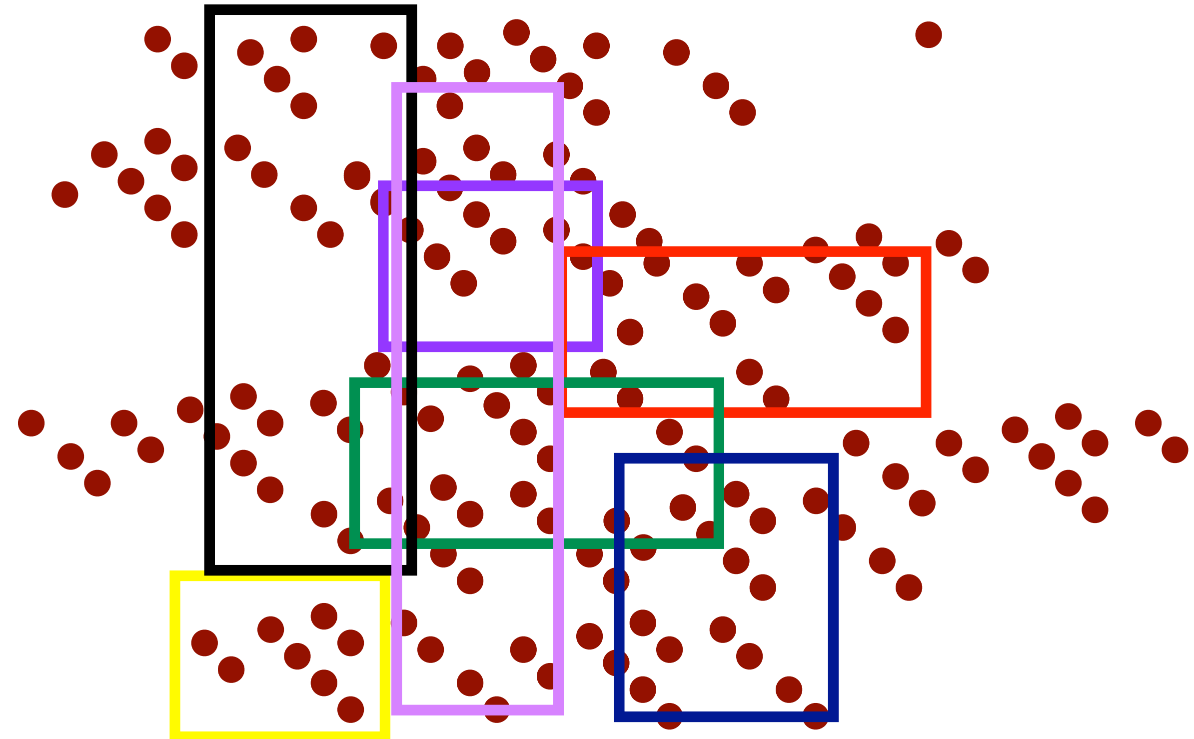


Solution:

S_3, S_{20}

$Insert(S_{10})$ $Insert(S_3)$ $Insert(S_3)$ $Insert(S_{20})$ $Delete(S_{10})$

Dynamic model



- In the **dynamic** model, we process **updates** and maintain an approximate solution efficiently.
- The main **constraint** of a dynamic algorithm is the **update** time.

- The **update** time is the number of queries that we ask to compute a solution $\mathcal{S}_t$ at time t given solution $\mathcal{S}_{t-1}$ at time $t-1$.

Main Theorem

There is an algorithm for dynamic submodular cover that

- Maintains an expected $(1-\epsilon, \epsilon^{-1})$-bicriteria solution
- Having expected amortized $poly(\log(n), \log(\rho), \epsilon^{-1})$ update time

$n = |V|$

Weight ratio $\rho = \dfrac{\max_{v \in V} w(v)}{\min_{v \in V} w(v)}$

Related work

Offline: **Wolsey [Combinatorica, 1982]** shows that greedy algorithm is a logarithmic approximation algorithm.

Streaming: **Norouzi-Fard et al. [NeurIPS, 2016]** give $(1-\epsilon, O(\epsilon^{-1}))$-bicriteria approximation algorithm for special case where the weights are uniform, i.e., each element has weight 1.

Related work

Dynamic: **Gupta and Levin [FOCS, 2020]** consider a different variant of the problem in which the submodular function f changes over time.

In contrast, our approach assumes that the underlying function is fixed and the ground set changes. This is aligned with the models considered in the streaming setting **[Norouzi-Fard et al., NeurIPS, 2016]**

Overview

Offline Algorithm

$L_0 = V$

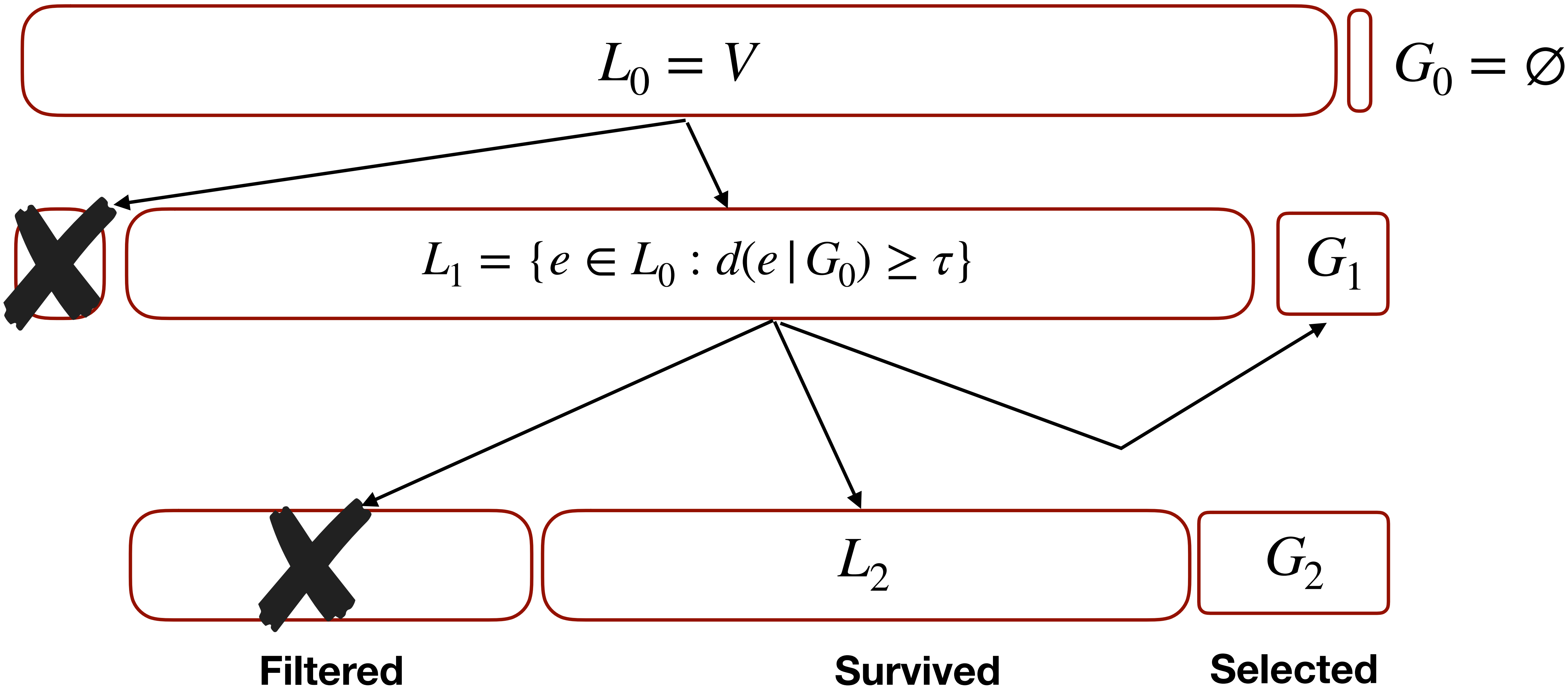
$G_0 = \varnothing$

Offline Algorithm

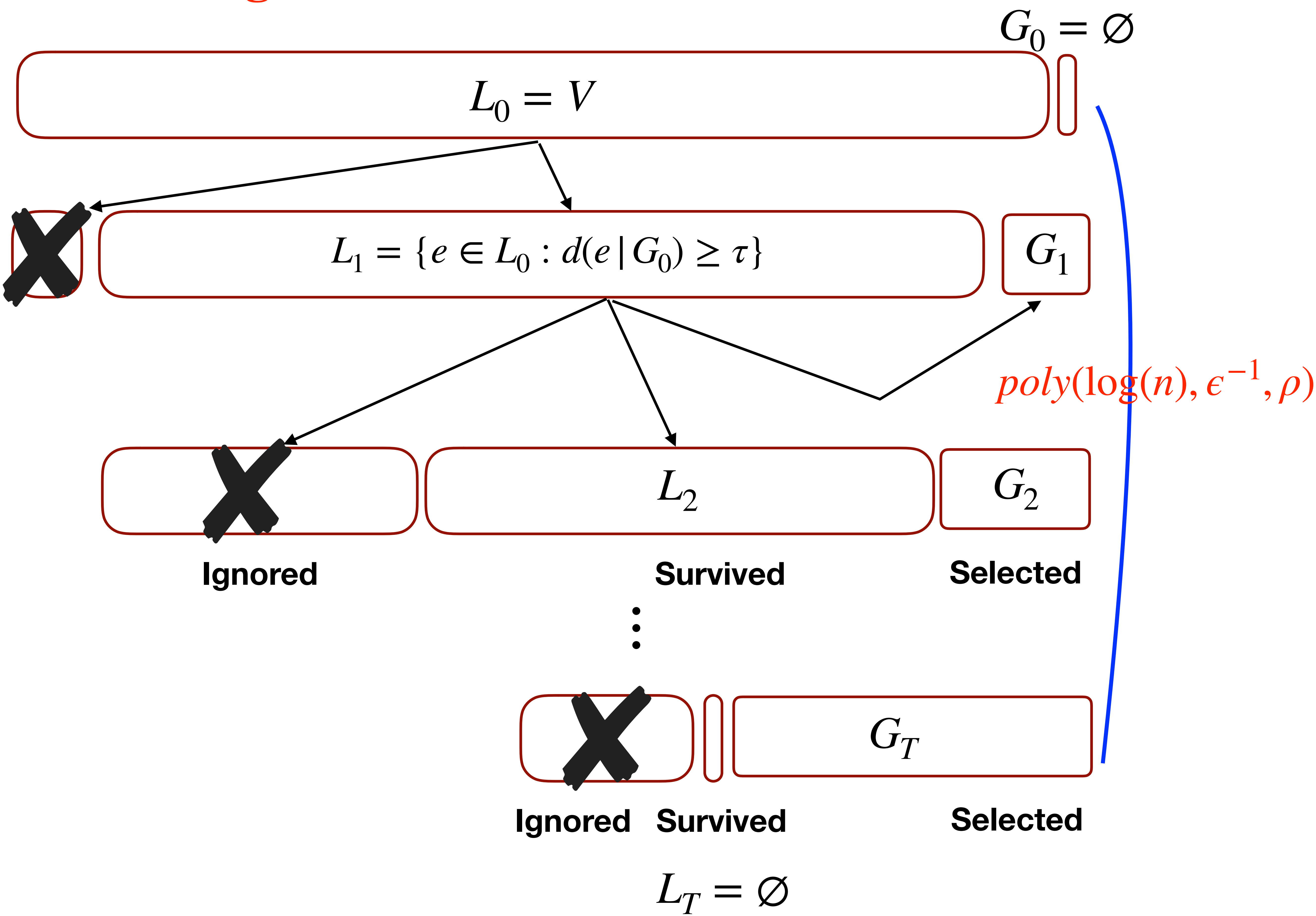
$$L_0 = V \qquad G_0 = \varnothing$$

$$L_1 = \{e \in L_0 : d(e \,|\, G_0) \geq \tau\} \qquad G_1$$

Offline Algorithm



Offline Algorithm



Sampling

At any level L_i, we bucketize elements based on **their marginal density** and weights and select a sample set S_i such that

- **Expansion(G_i):** In expectation, at least $(1-\epsilon)$-fraction of S_i is added to G_i
- **Filtering(V_i):** In expectation, at least $\dfrac{1}{poly(\log(n), \epsilon^{-1}, \rho)}$-fraction of elements of L_i are have their marginal gain decreased sufficiently and do not appear in L_{i+1}.

Thank you