Non-monotone Submodular Maximization with Nearly Optimal Adaptivity and Query Complexity

Matthew Fahrbach[1] Vahab Mirrokni[2] Morteza Zadimoghaddam[2]

June 13, 2019

[1]Georgia Tech [2]Google

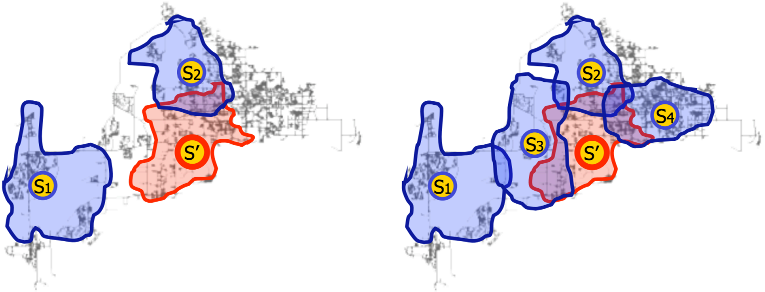
Submodular Functions

Def. A function $f : 2^N \to \mathbb{R}$ is submodular if for all $S \subseteq T \subseteq N$ and $x \in N \setminus T$ we have

$$f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T).$$

- Models the property of diminishing returns

Example. $f(S)$ is the coverage of placing sensors at locations S.



Submodular Functions

Def. A function $f : 2^N \to \mathbb{R}$ is submodular if for all $S \subseteq T \subseteq N$ and $x \in N \setminus T$ we have

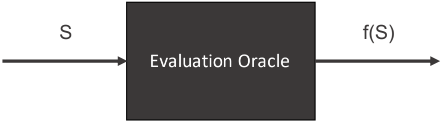
$$f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T).$$

- Models the property of diminishing returns

Applications in machine learning.

- Document summarization
- Exemplar clustering
- Feature selection
- Graph cuts

Assumption. Evaluation oracle that returns $f(S)$ in $O(1)$ time.



Problem. Maximize $f(S)$ such that $|S| \leq k$ using a small number of adaptive rounds and oracle queries.

Adaptivity Complexity

Def. The adaptivity of a distributed algorithm is the minimum needed round complexity, where in each round the algorithm can make poly(n) independent queries to the value oracle.

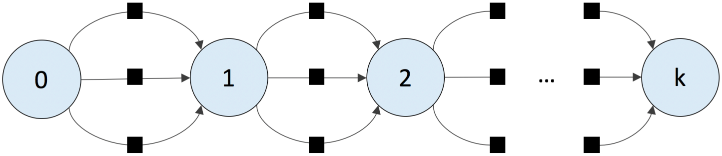
- Rank of partial order on queries ordered by dependence
- Models communication complexity with oracle

Adaptivity Complexity

Def. The adaptivity of a distributed algorithm is the minimum needed round complexity, where in each round the algorithm can make poly(n) independent queries to the value oracle.

Example. Greedy algorithm for constrained maximization

1. Set $S_0 \leftarrow \emptyset$
2. For $i = 1$ to k:
3. Set $S_i \leftarrow S_{i-1} \cup \{\arg\max_{x \in N} f(S_{i-1} \cup \{x\})\}$



Main Results

Problem. Submodular maximization of a non-monotone function subject to a cardinality constraint k

Algorithm	Approximation	Adaptivity	Queries
BFS16	$1/e \approx 0.371$	$O(k)$	$O(n)$
BBS18 (NeurIPS 18)	0.183	$O(\log^2(n))$	$\tilde{O}(\mathrm{OPT}^2 n)$
CQ19 (STOC 19)	0.172	$O(\log^2(n))$	$\tilde{O}(nk^4)$
ENV19 (STOC 19)	0.371	$O(\log^2(n))$	$\tilde{O}(nk^2)$
FMZ19	0.039	$O(\log(n))$	$O(n \log(k))$

Adaptivity Hardness [BS18]. We need $\Omega(\log(n)/\log\log(n))$ adaptive rounds to achieve a constant-factor approximation.