Randomized Algorithms for Submodular Function Maximization with a k-System Constraint

Shuang Cui [1], Kai Han *[1], Tianshuai Zhu [1], Jing Tang [2], Benwei Wu [1], He Huang [3]

*Correspondence to: hankai@ustc.edu.cn

[1]School of Computer Science and Technology, University of Science and Technology of China, [2]Data Science and Analytics Thrust, The Hong Kong University of Science and Technology, [3]School of Computer Science and Technology, Soochow University

Problem Definition

- Let $f: 2^N \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function, and let (N, I) be a k-system. The problem of submodular maximization with a k-system constraint can be formalized as:

$$\max\{f(S): S \in I\}$$

  - Non-adaptive setting

    $f(\cdot)$ is assumed to be deterministic.

  - Adaptive setting

    $f(\cdot)$ is assumed to be stochastic. The goal is to find a adaptive policy π that maximizes the expected utility of policy π.

Contributions

Algorithms	Source	Ratio	Time Complexity	Adaptive?
RepeatedGreedy	(Gupta et al., 2010)	$3k+6+3k^{-1}$	$\mathcal{O}(nrk)$	×
RepeatedGreedy	(Mirzasoleiman et al., 2016)	$2k+3+k^{-1}$	$\mathcal{O}(nrk)$	×
TwinGreedyFast	(Han et al., 2020)	$2k+2+\epsilon$	$\mathcal{O}(\frac{n}{\epsilon}\log(\frac{r}{\epsilon}))$	×
RepeatedGreedy	(Feldman et al., 2017)	$k+2\sqrt{k}+3+\frac{6}{\sqrt{k}}$	$\mathcal{O}(nr\sqrt{k})$	×
FastSGS	(Feldman et al., 2020)	$(1-2\epsilon)^{-2}(k+2\sqrt{k+2}+3)$	$\mathcal{O}(\frac{kn}{\epsilon}\log(\frac{n}{\epsilon}))$	×
RandomMultiGreedy	this work	$(1+\epsilon)(k+2\sqrt{k}+1)$	$\mathcal{O}(\frac{n}{\epsilon}\log(\frac{r}{\epsilon}))$	×
AdaptRandomGreedy	this work	$k+2\sqrt{k+1}+2$	$\mathcal{O}(nr)$	√

Approximation for submodular function maximization with a k-system constraint

➢ r is the rank of the considered k-system

- For the problem of submodular function maximization with a k-system constraint:
  - Under the non-adaptive setting

    we present the first randomized algorithm, which outperforms the state-of-the-art algorithm in (Feldman et al., 2020) in terms of both approximation ratio and time complexity.

  - Under the adaptive setting

    we present a randomized policy, which is the first adaptive algorithm to achieve a provable performance ratio when the utility function is not adaptive monotone.

Non-Adaptive Algorithm

Design of RandomMultiGreedy

- Iterate for T steps to construct ℓ disjoint candidate solutions $S_1, \ldots, S_\ell$.
- At each step t, greedily find a pair $(u_t, S_{i_t}) \in N \times [\ell]$ such that $S_{i_t} \cup \{u_t\} \in I$ and $f(u_t|S_{i_t})$ is maximized.

  - If $f(u_t|S_{i_t}) > 0$, add u_t into S_{i_t} with probability p.

  Remove u_t from N.
- The iterations stop immediately when the pair (u_t, S_{i_t}) cannot be found or $f(u_t|S_{i_t}) \leq 0$. Then RandomMultiGreedy returns S^* (which is the best one among $S_1, \ldots, S_\ell$).

Acceration

- Implement RandomMultiGreedy using a "lazy evaluation" method inspired by (Minoux, 1978; Ene & Nguyen, 2019).

The "Power of Randomization"

- By randomly discarding the elements with maximum marginal gain, the "local optima trap" problem can be mitigated for non-monotone submodular maximization.
- With the help of randomization, the time complexity of the RandomMultiGreedy is independent of k, due to the reason that it maintains only two candidate solutions. In contrast, the time complexities of all existing algorithms increase with k, as they maintain at least $\Omega(k^{0.5})$ candidate solutions.

  ➢ The following theorem reveals that the approximation ratio of RandomMultiGreedy is determined by ℓ and p, which allows us to set $\ell = 2$, p=2/(1+ $k^{0.5}$) to optimize the approximation ratio.

■ **Theorem**: For any $\ell \geq 2$ and $p \in (0,1]$, the RandomMultiGreedy(ℓ, p) algorithm outputs a solution S^* satisfying $f(O) \leq \frac{\ell\left(k+\frac{\ell}{p}-1\right)}{\ell-p}\mathbb{E}[f(S^*)]$.

Adaptive Algorithm

Design of AdaptRandomGreedy

- AdaptRandomGreedy runs in iterations and identify an element u^* in each iteration which maximizes the expected marginal gain $\Delta(u^* | \psi)$ without violating the feasibility I.

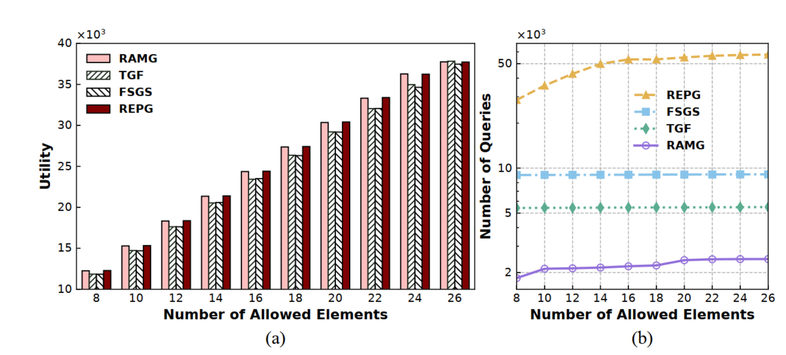
  - If $\Delta(u^* | \psi) > 0$, π_A observes the state of u^* and adds u^* into the solution set S with probability p.

  Remove $u^*$from N.
- The iterations stop immediately when no more elements can be added into S without violating the feasibility of I or $\Delta(u^* | \psi)$ is non-positive. Then RandomMultiGreedy returns S.

➢ π_A: the adaptive policy adopted by AdaptRandomGreedy
ψ: the partial realization observed by π_A at the moment that u^* is identified

■ **Theorem**: AdaptRandomGreedy achieves an approximation ratio of $\frac{pk+1}{p(1-p)}$ (i.e., $f_{avg}(\pi_A) \geq \frac{p(1-p)}{pk+1} \cdot f_{avg}(\pi_{opt})$) under time complexity of $O(nr)$. The ratio is minimized to $\left(1+\sqrt{k+1}\right)^2$ when $p = \left(1+\sqrt{k+1}\right)^{-1}$.

Performance Evaluation



Movie recommendation

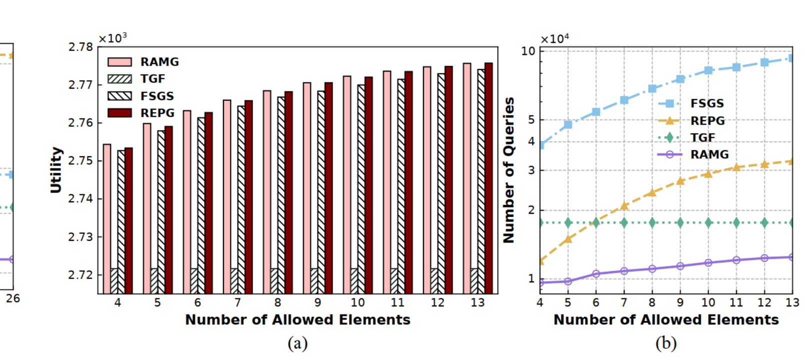
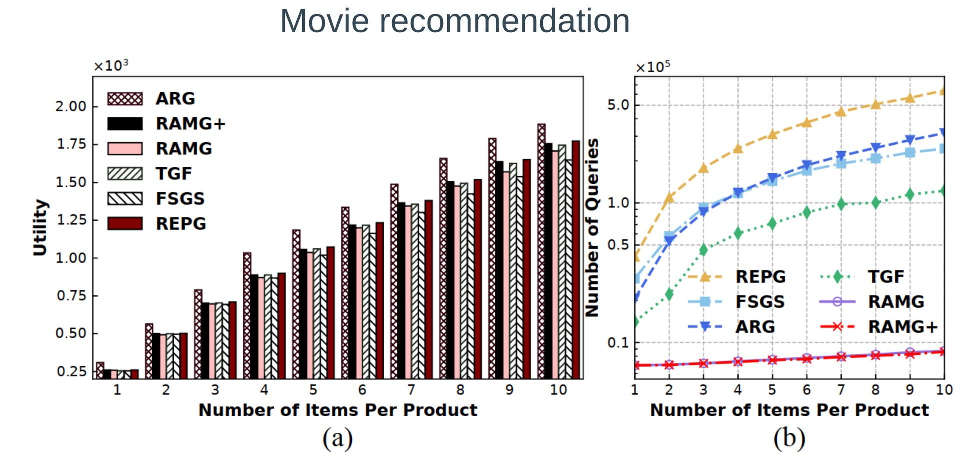


Image summarization



Social Advertising with Multiple Products

REPG : RepeatedGreedy
TGF : TwinGreedyFast
FSGS : FastFGS
RAMG,RAMG+: RandomMultiGreedy
ARG: AdaptRandomGreedy