Submodular Maximization Under the Intersection of Matroid and Knapsack Constraints

Yu-Ran Gu, Chao Bian, Chao Qian

LAMDA Group, Nanjing University {guyr,bianc,qianc}@lamda.nju.edu.cn

LAMDA Learning And Mining from DatA http://lamda.nju.edu.cn

The Problem

- **Submodular Maximization Problem (SMP)**: Given a finite set $\mathcal{N}$ and a submodular objective function $f: 2^{\mathcal{N}} \to \mathrm{R}^+$, to find $\arg\max_{S \subseteq \mathcal{N}} f(S)$
- **SMP under Knapsack and Matroid Constraints**: Given a finite set $\mathcal{N}$ and a submodular objective function $f: 2^{\mathcal{N}} \to \mathrm{R}^+$, a m-knapsack constraint with cost functions $c_1, \cdots, c_m$, and a k-matroid $\mathcal{M}(\mathcal{N}, \cap_{i=1}^{k} \mathcal{I}_i)$, to find $arg\ max_{S \subseteq \mathcal{N}}\ f(S)$ such that $S \in \cap_{i=1}^{k} \mathcal{I}_i$ and $\forall i \in [m], c_i(S) \leq 1$.

Comparison of the state-of-the-art algorithms:

Algorithm	Approximation	Running Time
FANTOM [Mirzasoleiman et al., ICML'16]	$(1+\epsilon)(2k+(2+2/k)m + O(1)$	$\tilde{O}(n^2/\epsilon)$
DENSITYSEARCHSGS [Feldman et al., arXiv'20]	$(1+\epsilon)(k+2m)+O(\sqrt{m})$	$\tilde{O}(n/\epsilon)$
SPROUT(This Paper)	$(1+\epsilon)(k+m)+O(\sqrt{m})$	$\tilde{O}(n^2/\epsilon)$

matroid: (1) $\emptyset \in \mathcal{I}$; (2) $\forall A \subseteq B \subseteq \mathcal{N}$, if $B \in \mathcal{I}$ then $A \in \mathcal{I}$; (3) if $\forall A, B \in \mathcal{I}$ and $|A| < |B|$, there is $e \in B \setminus A$ such that $A \cup e \in \mathcal{I}$

knapsack: $c(S) \leq 1$ given a modular cost function c

NP-hard in general!

Submodularity: $\forall X \subseteq Y, v \notin Y: f(X \cup \{v\}) - f(X) \geq f(Y \cup \{v\}) - f(Y)$

The SPROUT Algorithm

Input: Objective function $f: 2^{\mathcal{N}} \to \mathbb{R}_+$, k matroids $\mathcal{M}_i(\mathcal{N}, \mathcal{I}_i)$ and m cost functions $c_i: \mathcal{N} \to \mathbb{R}_+$

Parameter: Error params δ, ϵ, correction params β, γ, enumeration param C and number ℓ of solutions

Output: A set S s.t. $S \in \bigcap_{i=1}^{k} \mathcal{I}_i$ and $\forall i \in [m], c_i(S) \leq 1$

1: **for** each feasible $\mathcal{A} \subseteq \mathcal{N}$ with C elements **do**
2: $\quad z_{\mathcal{A}}(S) \triangleq f(S|\mathcal{A})$.
3: $\quad \mathcal{N}' \triangleq \{e \in \mathcal{N} | e \notin \mathcal{A} \wedge C \cdot z_{\mathcal{A}}(e) \leq f(\mathcal{A})\}$.
4: $\quad \mathcal{M}_i'(\mathcal{N}', \mathcal{I}_i') \triangleq$ contraction of $\mathcal{M}_i(\mathcal{N}, \mathcal{I}_i)$ by $\mathcal{A}$.
5: $\quad \mathcal{I}' \triangleq \bigcap_{i=1}^{k} \mathcal{I}_i'$.
6: $\quad$ Decrease knapsack budgets by $c_i(\mathcal{A})$ and normalize each of them to 1.
7: $\quad$ Let $S_0 = \emptyset$, and $\mathcal{V}$ be the maximum $z_{\mathcal{A}}$ value of a single feasible element in $\mathcal{N}'$.
8: $\quad$ Let $b_1 = 1$ and $b_0 = \lceil \log |\mathcal{N}'|/\delta \rceil$.
9: $\quad$ **while** $|b_1 - b_0| > 1$ **do**
10: $\quad\quad \rho = \beta\mathcal{V}(1+\delta)^{\lfloor (b_1+b_0+1)/2 \rfloor} + \gamma f(\mathcal{A})/C$.
11: $\quad\quad S_K = \text{KNAPSACKSGS}(z_{\mathcal{A}}, \mathcal{N}', \mathcal{I}', \{c_i\}_{i=1}^{m}, \ell, \rho, \epsilon)$
12: $\quad\quad$ Add S_K to S_0.
13: $\quad\quad b_E = \lfloor (b_1 + b_0 + 1)/2 \rfloor$.
14: $\quad$ **end while**
15: $\quad S_{\mathcal{A}} = \arg\max_{S \in S_0} f(S)$
16: **end for**
17: $\mathcal{A}^* = \arg\max_{\mathcal{A}} f(\mathcal{A} \cup S_{\mathcal{A}})$ over all feasible $\mathcal{A} \subseteq \mathcal{N}$
18: **return** $\mathcal{A}^* \cup S_{\mathcal{A}^*}$

Partial Enumeration By Reducing Problem Instance: To be more robust in practice and achieve better guarantee

Incorporating a partial enumeration technique [Badanidiyuru et al., NeurIPS'20] into the simultaneous greedy framework [Feldman et al., arXiv'20]

➤ Subroutine from [Feldman et al., arXiv'20]

Indicator-based Binary Search: To find solution with approximately best guarantee

Indicates whether the knapsack constraints are violated in line 6 of KNAPSACKSGS

The SPROUT++ Algorithm

Input: Objective function $f: 2^{\mathcal{N}} \to \mathbb{R}_+$, k matroids $\mathcal{M}_i(\mathcal{N}, \mathcal{I}_i)$ and m cost functions $c_i: \mathcal{N} \to \mathbb{R}_+$

Parameter: Error params δ, ϵ, correction params β, γ, acceleration param α, smooth param μ, counter t_c and number ℓ of solutions

Output: A set S s.t. $S \in \bigcap_{i=1}^{k} \mathcal{I}_i$ and $\forall i \in [m], c_i(S) \leq 1$

Let e^* be the feasible element $e \in \mathcal{N}$ maximizing $f(e)$.
while $t_c > 0$ **do**
$\quad$ Randomly select a feasible single-element set $\mathcal{A} \subseteq \mathcal{N}$ never being chosen before.
$\quad$ **if** $f(\mathcal{A}) \geq (1-\alpha)f(e^*)$ **then**
$\quad\quad z_{\mathcal{A}}(S) \triangleq f(S|\mathcal{A})$.
$\quad\quad \mathcal{N}' \triangleq \{e \in \mathcal{N} | e \notin \mathcal{A}\}$.
$\quad\quad \mathcal{M}_i'(\mathcal{N}', \mathcal{I}_i') \triangleq$ contraction of $\mathcal{M}_i(\mathcal{N}, \mathcal{I}_i)$ by $\mathcal{A}$.
$\quad\quad \mathcal{I}' \triangleq \bigcap_{i=1}^{k} \mathcal{I}_i'$.
$\quad\quad$ Decrease knapsack budgets by $c_i(\mathcal{A})$ and normalize each of them to 1.
$\quad\quad$ Let $S_0 = \emptyset$, and $\mathcal{V}$ be the maximum $z_{\mathcal{A}}$ value of a single feasible element in $\mathcal{N}'$.
$\quad\quad$ Let $b_1 = 1$ and $b_0 = \lceil \log |\mathcal{N}'|/\delta \rceil$.
$\quad\quad$ **while** $|b_1 - b_0| > 1$ **do**
$\quad\quad\quad b = \lfloor (b_1 + b_0 + 1)/2 \rfloor$.
$\quad\quad\quad \rho = \beta\mathcal{V}(1+\delta)^{b} + \gamma f(\mathcal{A})$.
$\quad\quad\quad S_K = \text{KNAPSACKSGS}(z_{\mathcal{A}}, \mathcal{N}', \mathcal{I}', \{c_i\}_{i=1}^{m}, \ell, \rho, \epsilon)$.
$\quad\quad\quad$ Add S_K to S_0.
$\quad\quad\quad b_E = b + (1-2E)(1-1/\mu)|b_E - b|$.
$\quad\quad$ **end while**
$\quad\quad S_{\mathcal{A}} = \arg\max_{S \in S_0} f(S)$.
$\quad\quad t_c = t_c - 1$.
$\quad$ **end if**
end while
$\mathcal{A}^* = \arg\max_{\mathcal{A}} f(\mathcal{A} \cup S_{\mathcal{A}})$ over all feasible $\mathcal{A} \subseteq \mathcal{N}$.
return $\mathcal{A}^* \cup S_{\mathcal{A}^*}$.

Random Sampling and Threshold Filtering: To get more valuable elements more efficiently

Delete No Extra Elements: To maintain high-quality elements

Aim to BE MORE EFFICIENT!!!

Smooth Technique: To avoid search range shrinking so fast that many good solutions may be missed

Smooth Parameter: Adjust the step size of binary search

Theoretical Analysis

- SPROUT can achieve the SOTA approximation guarantee BETTER THAN PREVIOUS ALGORITHMS

Theorem 1. SPROUT achieves an approximation ratio of **roughly** $\left(\frac{1-\epsilon}{k+m+3+2\sqrt{m+1}} + \frac{(1-\epsilon)C}{r}\right)^{-1}$ **using** $\tilde{O}\left(\frac{Pn^{C+1}}{\epsilon}\right)$ **oracle calls and** $\tilde{O}\left(\frac{Pmn^{C+1}}{\epsilon}\right)$ **arithmetic operations**, where $P = \{\lceil\sqrt{1+m}\rceil, k\}$ and r is the size of S_{OPT}.

Lemma 1. In SPROUT, $f(\mathcal{A} \cup S_K) \geq \min\{\rho + \left(1 - \frac{1}{C}\right)f(\mathcal{A}), \frac{(1-\epsilon)}{p+1}\left(\left(1 - \frac{1}{\ell} - \epsilon\right)z_{\mathcal{A}}(S'_{OPT}) - \rho m\right) + f(\mathcal{A})\}$ for each generated ρ in line 10 and corresponding S_K, where S'_{OPT} refers to an optimal solution for the reduced instance, and $p = \max\{\ell - 1, k\}$.

Proof sketch: Using Lemma 1 $\Longrightarrow$ $f(S) \geq \min\{\rho + \left(1 - \frac{1}{C}\right)f(\mathcal{A}), \frac{(1-\epsilon)}{p+1}\left(\left(1 - \frac{1}{\ell} - \epsilon\right)z_{\mathcal{A}}(S'_{OPT}) - \rho m\right) + f(\mathcal{A})\}$ $\Longrightarrow$ Best density ratio ρ^*: $\rho^* = \dfrac{(1-\epsilon)\left(1 - \frac{1}{\ell} - \epsilon\right)z_{\mathcal{A}}(S'_{OPT}) + \frac{(p+1)f(\mathcal{A})}{C}}{p+1+m(1-\epsilon)}$

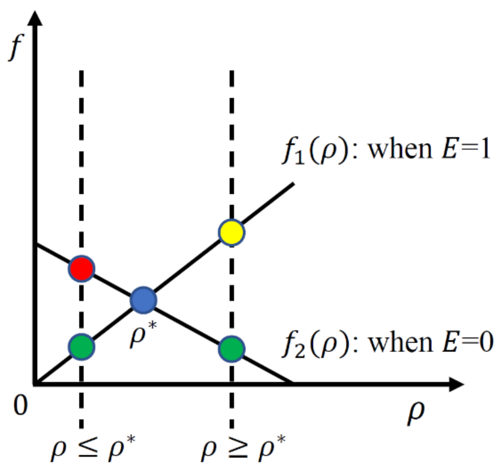
To find ρ' s.t. $(1-\delta)\rho^* \leq \rho' \leq \rho^*$ ⇩

- SPROUT++ can achieve a SIMILAR APPROXIMATION GUARANTEE to SPROUT with a high probability using MUCH LESS TIME under an assumption

Theorem 2. Suppose that $\forall a \in S_{OPT}, (1+\alpha)f(a) \geq f(e^*)$, where e^* is a feasible max-value element in $\mathcal{N}$ and $\alpha \leq \frac{(1-\epsilon)(p+1-(1-\epsilon)^2)}{\epsilon(p+1)+m(1-\epsilon)}$. SPROUT++ offers an approximation ratio of $(1+\epsilon)(k+m+3+2\sqrt{m+1})$ with probability at least $1 - e^{\frac{-rt_c}{n}}$ **using** $\tilde{O}\left(\frac{\log^{-1}\{\frac{2\mu}{2\mu-1}\}t_c P n}{\epsilon}\right)$ **oracle calls and** $\tilde{O}\left(\frac{\log^{-1}\{\frac{2\mu}{2\mu-1}\}t_c P m n}{\epsilon}\right)$ **arithmetic operations.**

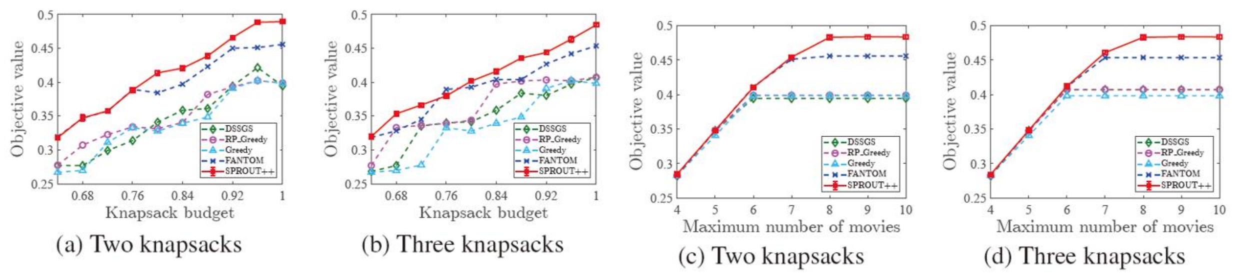
The objective value of each element in S_{OPT} is relatively large, which can hold if the marginal gain of adding each element e to $S_{OPT} \setminus e$ is large enough by the submodularity.

1) In one iteration of search, $\rho \leq \rho^*$ and $E = 0$.(red)
2) In one iteration of search, $\rho \geq \rho^*$ and $E = 1$.(yellow)
3) $\rho \leq \rho^*$ implies E = 1, then increase b_1 to $\lfloor \frac{b1+b0+1}{2} \rfloor$; $\rho \geq \rho^*$ implies $E = 0$, then decrease b_0 to $\lfloor \frac{b_1+b_0+1}{2} \rfloor$. (green)



Empirical Studies

Movie recommendation



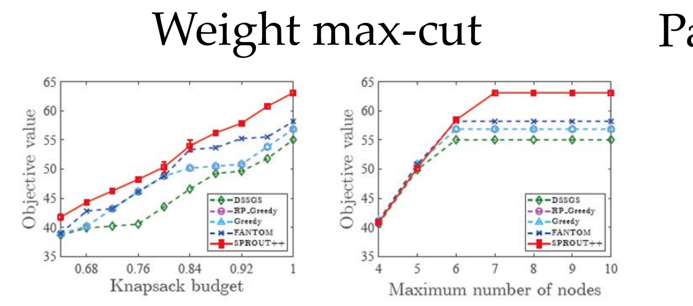
(a) Two knapsacks (b) Three knapsacks

obj. value vs. knapsack budget

(c) Two knapsacks (d) Three knapsacks

obj. value vs. maximum number of movies

Weight max-cut

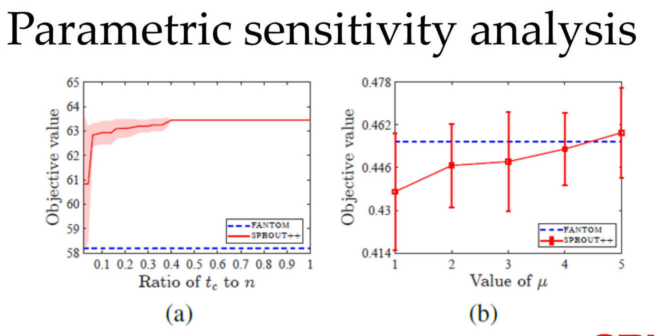


(a) (b)

obj. value vs. knapsack budget

obj. value vs. maximum number of nodes

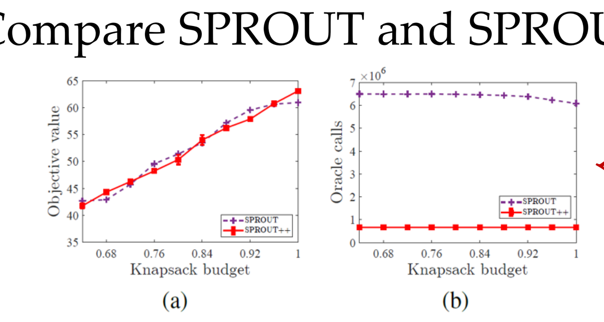
Parametric sensitivity analysis



(a) (b)

t_c and μ are useful for efficiency

Compare SPROUT and SPROUT++



(a) (b)

SPROUT++ can compete with SPROUT while being much faster!

SPROUT++: ALWAYS THE BEST!

Conclusion

- Propose SPROUT algorithm with the **SOTA approximation ratio** for submodular maximization under the intersection of matroid and knapsack constraints and propose SPROUT++ algorithm to **improve the efficiency**
- **Demonstrate the superior performance** of SPROUT & SPROUT++ in practice by extensive experiments