Maximum Coverage over a Matroid Constraint

Yuval Filmus   Justin Ward
University of Toronto

STACS 2012, Paris

Max Coverage: History

- Location of bank accounts: Cornuejols, Fisher & Nemhauser 1977, Management Science
- Official definition: Hochbaum & Pathria 1998, Naval Research Quarterly
- Lower bound: Feige 1998
- Extended to Submodular Max. over a Matroid: Calinescu, Chekuri, Pál & Vondrák 2008 (with help from Ageev & Sviridenko 2004)

We consider Maximum Coverage over a Matroid.

Maximum Coverage ...

Input:

- Universe U with weights $w \geq 0$
- Sets $S_i \subset U$
- Number n

Goal:

- Find n sets S_i that maximize $w(S_{i_1} \cup \cdots \cup S_{i_n})$

Maximum Coverage ...

Input:

- Universe U with weights $w \geq 0$
- Sets $S_i \subset U$
- Number n

Goal:

- Find n sets S_i that maximize $w(S_{i_1} \cup \cdots \cup S_{i_n})$

Greedy algorithm gives $1 - 1/e$ approximation.

Feige ('98): optimal unless P=NP.

... over a Matroid

Input:

- Universe U with weights $w \geq 0$
- Sets $S_i \subset U$
- Matroid $\mathfrak{m}$ over set of all S_i

Goal:

- Find collection of sets $\mathcal{S} \in \mathfrak{m}$ that maximizes $w(\bigcup \mathcal{S})$

What is a matroid?

Invented by Whitney (1935).

Definition: Matroid

A collection of *independent sets* s.t.

1. A independent, $B \subset A \Rightarrow B$ independent.
2. A, B independent, $|A| > |B| \Rightarrow$ there exists some $x \in A \setminus B$ s.t. $B \cup x$ is independent.

What is a matroid?

Invented by Whitney (1935).

Definition: Matroid

A collection of *independent sets* s.t.

1. A independent, $B \subset A \Rightarrow B$ independent.
2. A, B independent, $|A| > |B| \Rightarrow$ there exists some $x \in A \setminus B$ s.t. $B \cup x$ is independent.

Partition Matroid

- $\mathcal{F}_1, \ldots, \mathcal{F}_n$ *disjoint* sets.
- Independent set: ≤ 1 set from each $\mathcal{F}_i$.

Max Coverage over a Partition Matroid

Input:

- Universe U with weights $w \geq 0$
- n families $\mathcal{F}_i \subset 2^U$

Goal:

- Find collection of sets $S_i \in \mathcal{F}_i$ that maximizes $w(S_1 \cup \cdots \cup S_n)$

Some algorithms

Greedy

1. Pick set S_1 of maximal weight.
2. Pick set S_2 of maximal *additional* weight.
3. And so on.

Some algorithms

Greedy

1. Pick set S_1 of maximal weight.
2. Pick set S_2 of maximal *additional* weight.
3. And so on.

Local Search

1. Start at some solution $S_1, \ldots, S_n$.
2. Replace some S_i with some S_i' that improves total weight.
3. Repeat Step 2 while possible.

Failure of greedy

Bad instance for Greedy

$$\frac{\begin{array}{c} A_1 = \{x, \epsilon\} \\ A_2 = \{y\} \end{array}}{\mathcal{F}_1} \qquad \frac{B = \{x\}}{\mathcal{F}_2}$$

$$w(x) = w(y) \gg w(\epsilon)$$

Greedy chooses $\{A_1, B\}$, optimal is $\{A_2, B\}$.

Resulting approximation ratio is only $1/2$.

Failure of greedy

Bad instance for Greedy

$$\frac{\begin{array}{c} A_1 = \{x, \epsilon\} \\ A_2 = \{y\} \end{array}}{\mathcal{F}_1} \qquad \frac{B = \{x\}}{\mathcal{F}_2}$$

$$w(x) = w(y) \gg w(\epsilon)$$

Greedy chooses $\{A_1, B\}$, optimal is $\{A_2, B\}$.

Resulting approximation ratio is only $1/2$.

Local search finds optimal solution.

Maybe local search?

Bad instance for Local Search

$$\frac{A_1 = \{x, \epsilon_x\} \quad A_2 = \{y\}}{\mathcal{F}_1} \qquad \frac{B_1 = \{x\} \quad B_2 = \{\epsilon_y\}}{\mathcal{F}_2}$$

$$w(x) = w(y) \gg w(\epsilon_x) = w(\epsilon_y)$$

$\{A_1, B_2\}$ is local maximum. Optimum is $\{A_2, B_1\}$.

Maybe local search?

Bad instance for Local Search

$$\frac{\begin{array}{c} A_1 = \{x, \epsilon_x\} \\ A_2 = \{y\} \end{array}}{\mathcal{F}_1} \qquad \frac{\begin{array}{c} B_1 = \{x\} \\ B_2 = \{\epsilon_y\} \end{array}}{\mathcal{F}_2}$$

$$w(x) = w(y) \gg w(\epsilon_x) = w(\epsilon_y)$$

$\{A_1, B_2\}$ is local maximum. Optimum is $\{A_2, B_1\}$.

k-local search (on SBO matroids) has approx ratio

$$\frac{1}{2} + \frac{k-1}{2n-k-1}.$$

Local search fantasy

$$\begin{array}{cc} A_1 = \{x, \epsilon_x\} & B_1 = \{x\} \\ A_2 = \{y\} & B_2 = \{\epsilon_y\} \\ \hline \end{array}$$

Fantasy algorithm

Local search fantasy

$$A_1 = \{x, \epsilon_x\} \quad B_1 = \{x\}$$
$$A_2 = \{y\} \quad B_2 = \{\epsilon_y\}$$

$$x \times 1$$
$$\epsilon_x \times 1$$

Greedy stage

Local search fantasy

$$A_1 = \{x, \epsilon_x\} \quad B_1 = \{x\}$$
$$A_2 = \{y\} \quad B_2 = \{\epsilon_y\}$$

$$x \times 1$$
$$\epsilon_x \times 1$$
$$\epsilon_y \times 1$$

Greedy stage

Local search fantasy

$$\begin{array}{cc} A_1 = \{x, \epsilon_x\} & B_1 = \{x\} \\ A_2 = \{y\} & B_2 = \{\epsilon_y\} \\ \hline \multicolumn{2}{c}{x \times 2} \\ \multicolumn{2}{c}{\epsilon_x \times 1} \end{array}$$

Local search stage

We lose ϵ_y but gain second appearance of x.

Local search fantasy

$$A_1 = \{x, \epsilon_x\} \quad B_1 = \{x\}$$
$$A_2 = \{y\} \quad B_2 = \{\epsilon_y\}$$

$$x \times 1$$
$$y \times 1$$

Local search stage

Local search fantasy

$$\frac{\begin{array}{cc} A_1 = \{x, \epsilon_x\} & B_1 = \{x\} \\ A_2 = \{y\} & B_2 = \{\epsilon_y\} \end{array}}{\begin{array}{c} x \times 1 \\ y \times 1 \end{array}}$$

Done — found optimal solution

Non-oblivious local search

Idea

Give more weight to duplicate elements.

Use local search with auxiliary objective function (Alimonti '94; Khanna, Motwani, Sudan & U. Vazirani '98):

$$f(\mathcal{S}) = \sum_{u \in U} \alpha_{\#_u(\mathcal{S})} w(u).$$

Change is considered beneficial if it improves $f(\mathcal{S})$.

Oblivious local search: $\alpha_0 = 0$, $\alpha_i = 1$ for $i \geq 1$.

Choosing the weights

Consider what happens at termination.

Setup:

- $S_1, \ldots, S_n$: local maximum.
- $O_1, \ldots, O_n$: optimal solution.

Local optimality implies

$$nf(S_1, \ldots, S_n) \geq \sum_{i=1}^{n} f(S_1, \ldots, S_{i-1}, O_i, S_{i+1}, \ldots, S_n)$$

Choosing the weights

Parametrize situation using $w_{l,c,g}$ = total weight of elements which belong to

- $l + c$ sets S_i
- $g + c$ sets O_i
- c of the indices in common

Each element of the universe occurs in some $w_{l,c,g}$.

Choosing the weights

Local optimality implies

$$nf(S_1, \ldots, S_n) \geq \sum_{i=1}^{n} f(S_1, \ldots, S_{i-1}, O_i, S_{i+1}, \ldots, S_n)$$

In terms of $w_{l,c,g}$, this is

$$\sum_{l,c,g} \left[l(\alpha_{l+c} - \alpha_{l+c-1}) + g(\alpha_{l+c} - \alpha_{l+c+1}) \right] w_{l,c,g} \geq 0$$

Choosing the weights

Local optimality translates to

$$\sum_{l,c,g} \left[(l+g)\alpha_{l+c} - l\alpha_{l+c-1} - g\alpha_{l+c+1}\right] w_{l,c,g} \geq 0$$

Also,

$$w(O_1, \ldots, O_n) = \sum_{g+c \geq 1} w_{l,c,g}$$

$$w(S_1, \ldots, S_n) = \sum_{l+c \geq 1} w_{l,c,g}$$

Choosing the weights

Approximation ratio θ is given by

$$\max_{\alpha_i} \min_{w_{l,c,g}} w(S_1, \ldots, S_n)$$

s.t.

$$w(O_1, \ldots, O_n) = 1$$

$$nf(S_1, \ldots, S_n) \geq \sum_{i=1}^{n} f(S_1, \ldots, S_{i-1}, O_i, S_{i+1}, \ldots, S_n)$$

$$w_{l,c,g} \geq 0$$

Choosing the weights

Approximation ratio θ is given by

$$\max_{\alpha_i} \min_{w_{l,c,g}} \sum_{l+c\geq 1} w_{l,c,g}$$

$s.t.$

$$\sum_{g+c\geq 1} w_{l,c,g} = 1$$

$$\sum_{l,c,g} \left[(l+g)\alpha_{l+c} - l\alpha_{l+c-1} - g\alpha_{l+c+1}\right] w_{l,c,g} \geq 0$$

$$w_{l,c,g} \geq 0$$

Choosing the weights

Dualize the inner LP:

$$\max_{\alpha_i,\theta} \theta$$

$s.t.$

$$l(\alpha_l - \alpha_{l-1}) \leq 1$$

$$-g\alpha_1 \leq -\theta$$

$$(l+g)\alpha_{l+c} - l\alpha_{l+c-1} - g\alpha_{l+c+1} \leq 1 - \theta$$

($c \geq 1$ or $l, g \geq 1$)

Optimal weights

Solution to LP is $\theta = 1 - 1/e$ and

$$\begin{aligned} \alpha_0 &= 0, \\ \alpha_1 &= \theta, \\ \alpha_{l+1} &= (l+1)\alpha_l - l\alpha_{l-1} - (1-\theta). \end{aligned}$$

Sequence monotone concave, $\alpha_l = \frac{1}{e}\log l + O(1)$.

Optimal weights

Solution to LP is $\theta = 1 - 1/e$ and

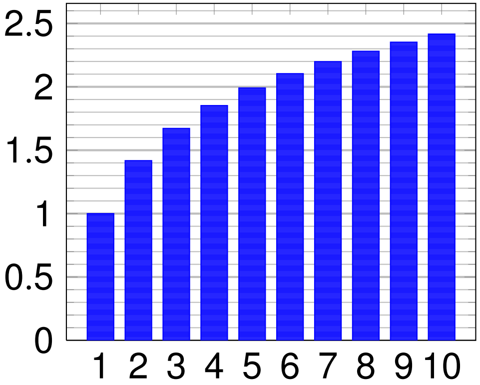
$$\begin{aligned} \alpha_0 &= 0, \\ \alpha_1 &= \theta, \\ \alpha_{l+1} &= (l+1)\alpha_l - l\alpha_{l-1} - (1-\theta). \end{aligned}$$

Sequence monotone concave, $\alpha_l = \frac{1}{e} \log l + O(1)$.

For rank n, can replace e with

$$e^{[n]} = \sum_{k=0}^{n-1} \frac{1}{k!} + \frac{1}{(n-1)\cdot(n-1)!} \approx e + \frac{1}{(n+2)!}.$$

Optimal weights (normalized)



Recap

Our Algorithm

1. Start with some solution $S_1, \ldots, S_n$.
2. Repeat while possible:
   Replace some S_i with some S_i' improving

$$f(S_1, \ldots, S_n) = \sum_{u \in S_1 \cup \cdots \cup S_n} \alpha_{\#_u(S_1,\ldots,S_n)} w(u).$$

Main Theorem

At the end of the algorithm,

$$w(S_1, \ldots, S_n) \geq \left(1 - \frac{1}{e}\right) w(O_1, \ldots, O_n).$$

Further work

Our framework generalizes.

Montone submodular maximization over a matroid

Optimal combinatorial algorithm.

Continuous algorithm by Calinescu, Chekuri, Pál and Vondrák (STOC 2008).

Further work

Our framework generalizes.

Montone submodular maximization over a matroid

Optimal combinatorial algorithm.

... with curvature constraint

Optimal combinatorial algorithm.
NP-hardness result (extending Feige 1998).

Vondrák 2010: extended continuous algorithm, gave lower bound in oracle model.

Further work

Our framework generalizes.

Montone submodular maximization over a matroid

Optimal combinatorial algorithm.

... with curvature constraint

Optimal combinatorial algorithm.

NP-hardness result .

Submodular maximization over bases of matroid

$1 - 2/e$ combinatorial algorithm.

Best prior result: $1/4$ (Vondrák, FOCS 2009).

Thank you

Questions?