# Maximum Coverage over a Matroid Constraint

Yuval Filmus  Justin Ward
University of Toronto

China Theory Week 2011

# 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: $\leq 1$ 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 $w(S_1 \cup \cdots \cup S_n)$

$n$ is the *rank* of the matroid.

# 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

$$\frac{A_1 = \{x, \epsilon\} \quad A_2 = \{y\}}{\mathcal{F}_1} \qquad \frac{B = \{x\}}{\mathcal{F}_2}$$

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

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

Resulting approximation ratio is only $1/2$.

# Failure of greedy

## Bad instance for Greedy

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

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

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

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

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

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

$k$-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

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

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

$$\frac{\begin{matrix} A_1 = \{x, \epsilon_x\} & B_1 = \{x\} \\ A_2 = \{y\} & B_2 = \{\epsilon_y\} \end{matrix}}{\begin{matrix} x \times 2 \\ \epsilon_x \times 1 \end{matrix}}$$

Local search stage

We lose $\epsilon_y$ 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 (using Brualdi's theorem)

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

# 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

I.e., up to permutation

$$S_1 \cap \cdots \cap S_l \cap S_{l+1} \cap \cdots \cap S_{l+c} \cap O_{l+1} \cap \cdots \cap O_{l+c} \cap O_{l+c+1} \cap \cdots \cap O_{l+c+g}$$

# Choosing the weights

Local optimality implies

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

# Choosing the weights

Local optimality implies

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

Take some $x$. Let $L = \{i : x \in S_i\}$, $G = \{i : x \in O_i\}$.

# Choosing the weights

Local optimality implies

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

Take some $x$. Let $L = \{i : x \in S_i\}$, $G = \{i : x \in O_i\}$.

- Coefficient on the left: $n\alpha_{|L|}$

# Choosing the weights

Local optimality implies

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

Take some $x$. Let $L = \{i : x \in S_i\}$, $G = \{i : x \in O_i\}$.

- Coefficient on the left: $n\alpha_{|L|}$
- Coefficient on the right:

$$(|\overline{L \cup G}| + |L \cap G|)\alpha_{|L|} + |L \setminus G| \cdot \alpha_{|L|-1} + |G \setminus L| \cdot \alpha_{|L|+1}$$

# Choosing the weights

Local optimality implies

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

Take some $x$. Let $L = \{i : x \in S_i\}$, $G = \{i : x \in O_i\}$.

- Coefficient on the left: $n\alpha_{|L|}$
- Coefficient on the right:

$$(|\overline{L \cup G}| + \underbrace{|L \cap G|}_{c})\alpha_{|L|} + \underbrace{|L \setminus G|}_{l} \cdot \alpha_{|L|-1} + \underbrace{|G \setminus L|}_{g} \cdot \alpha_{|L|+1}$$

So local optimality is equivalent to

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

# 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 $\theta$ 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 $\theta$ 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, fix second variable to 1:

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

$$s.t.$$

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

$$-g\alpha_1 \leq -\theta \quad {}_{g\geq 1}$$

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

$$c\geq 1 \text{ or } l,g\geq 1$$

Fold both max's to get an LP for the coefficients $\alpha_i$.

# 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

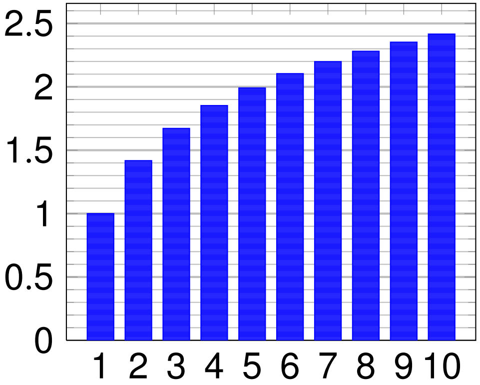
$$\begin{aligned} \alpha_0 &= 0, \\ \alpha_1 &= \theta, \\ \alpha_{l+1} &= (l+1)\alpha_l - l\alpha_{l-1} - (1-\theta). \end{aligned}$$

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)



# What now?

## Conclusion

Optimal combinatorial algorithm for maximum coverage over a matroid.

## Holy grail

Optimal combinatorial algorithm for monotone submodular maximization over a matroid.

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

Work in progress. We're hopeful.

# Monotone submodular functions

## Monotonicity

$$A \subseteq B \Rightarrow f(A) \leq f(B)$$

# Monotone submodular functions

## Monotonicity

$$A \subseteq B \Rightarrow f(A) \leq f(B)$$

## Submodularity

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$$

Discrete analog of *concave*.

# Monotone submodular functions

**Monotonicity**

$$A \subseteq B \Rightarrow f(A) \leq f(B)$$

**Submodularity**

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$$

Discrete analog of *concave*.

Coverage function is monotone submodular.

# Generalizing the algorithm

Original function depended on *elements*.

No longer have elements, so instead use

$$g(S) = \sum_{T \subseteq S} \beta_{|T|} f(T).$$

- $f$ is actual objective function (monotone submodular).
- $g$ is objective function used for local search.

# Choosing the coefficients

If $f$ is a coverage function, can recover elements using inclusion-exclusion, so can interpret previous algorithm in new setting.

Surprisingly, works for *general* monotone submodular functions up to rank 6.

# Choosing the coefficients

If $f$ is a coverage function, can recover elements using inclusion-exclusion, so can interpret previous algorithm in new setting.
Surprisingly, works for *general* monotone submodular functions up to rank 6.

Stops working at rank 7 (explicit counterexample).
Can calculate optimal coefficients as before.
Empirically, still yields $1 - 1/e$.
Work in progress.

# Conclusions

Our results:

- Combinatorial algorithm for maximum coverage over a matroid with optimal approximation ratio.
- Possible extension to arbitrary monotone submodular functions.

Questions?