From query complexity to computational complexity (for optimization of submodular functions)

Shahar Dobzinski[1] Jan Vondrák[2]

[1]Cornell University
Ithaca, NY
[2]IBM Almaden Research Center
San Jose, CA

Submodular optimization

Submodular functions:

1. A general framework capturing useful combinatorial structure
2. A natural property to be assumed in certain settings (utilities/valuations with diminishing returns)
3. A discrete variant of convexity / concavity

Recent interest: primarily due to

- Ability to unify/generalize previously studied problems
- Reappearing approximation factors ($\frac{1}{2}$, $1 - \frac{1}{e}$, ...) - where do they come from?
- Algorithmic game theory - submodular functions form an important class of valuations

Submodular Functions

Definition

A function $f : 2^X \to \mathbb{R}$ is submodular if for any S, T,

$$f(S \cup T) + f(S \cap T) \leq f(S) + f(T).$$

Submodular Functions

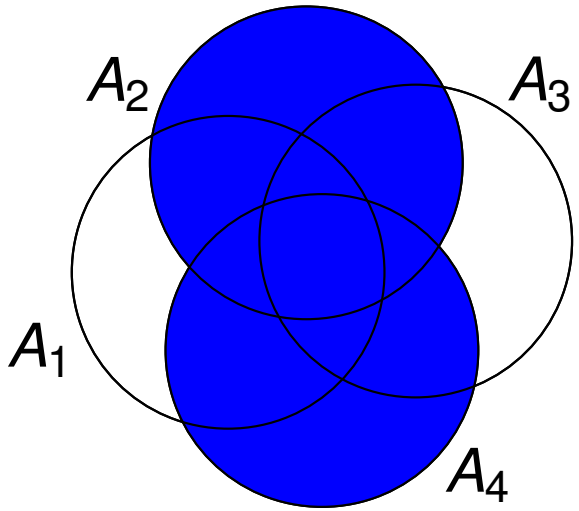
Definition

A function $f : 2^X \to \mathbb{R}$ is submodular if for any S, T,

$$f(S \cup T) + f(S \cap T) \leq f(S) + f(T).$$

Coverage function:
Given $A_1, \ldots, A_n \subset U$,

$$f(S) = \left| \bigcup_{j \in S} A_j \right|.$$



Submodular Functions

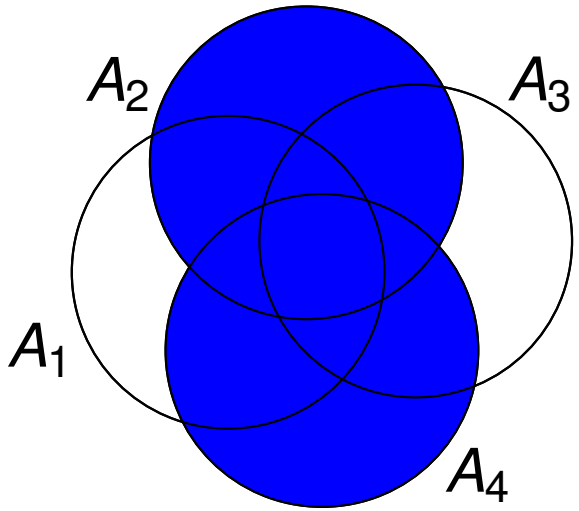
Definition

A function $f : 2^X \to \mathbb{R}$ is submodular if for any S, T,

$$f(S \cup T) + f(S \cap T) \leq f(S) + f(T).$$

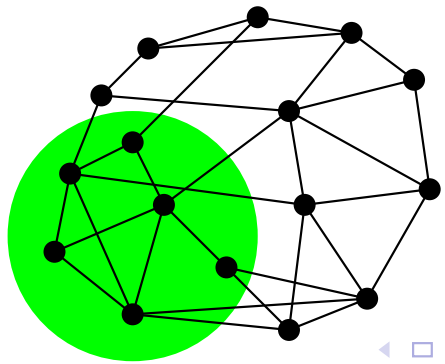
Coverage function:
Given $A_1, \ldots, A_n \subset U$,

$$f(S) = \left| \bigcup_{j \in S} A_j \right|.$$



Cut function:

$$\delta(T) = |e(T, \overline{T})|$$



Access models

In general, there are too many submodular functions ($2^{2^{\Omega(n)}}$) to describe them explicitly in poly(n) space.

Oracle models:

- *Value oracle:* For a given set S, what is $f(S) =$?
- *Demand oracle:* For given prices p_i, which set maximizes $f(S) - \sum_{i \in S} p_i$?

Access models

In general, there are too many submodular functions ($2^{2^{\Omega(n)}}$) to describe them explicitly in poly(n) space.

Oracle models:

- *Value oracle:* For a given set S, what is $f(S) =$?
- *Demand oracle:* For given prices p_i, which set maximizes $f(S) - \sum_{i \in S} p_i$?

Succinct representation:

- A string $a \in \{0,1\}^{poly(n)}$, encoding a function $f_a : 2^{[n]} \to \mathbb{R}$.
- The encoding should allow **efficient evaluation**: Given a and S, compute $f_a(S)$ in time $poly(n)$.

Example: cut functions have a natural succinct description.

Submodular maximization

Unconstrained submodular maximization:
given $f : 2^X \to \mathbb{R}_+$, *maximize* $f(S)$ *over all* $S \subseteq X$.
NP-hard, generalizes Max Cut.

Submodular maximization

Unconstrained submodular maximization:
given $f : 2^X \to \mathbb{R}_+$, *maximize* $f(S)$ *over all* $S \subseteq X$.
NP-hard, generalizes Max Cut.

Recent result: [Buchbinder,Feldman,Naor,Schwartz '12]

- $\frac{1}{2}$-approximation for unconstrained submodular maximization (following a line of improvements: $0.4 \Rightarrow 0.41 \Rightarrow 0.42 \Rightarrow 0.45 \ldots$)

We already knew: [Feige,Mirrokni,V. '07]

- $(\frac{1}{2} + \epsilon)$-approximation for unconstrained submodular maximization in the *value oracle model* would need exponentially many queries

Submodular maximization

Unconstrained submodular maximization:
given $f : 2^X \to \mathbb{R}_+$, *maximize* $f(S)$ *over all* $S \subseteq X$.
NP-hard, generalizes Max Cut.

Recent result: [Buchbinder,Feldman,Naor,Schwartz '12]

- $\frac{1}{2}$-approximation for unconstrained submodular maximization (following a line of improvements: $0.4 \Rightarrow 0.41 \Rightarrow 0.42 \Rightarrow 0.45 \ldots$)

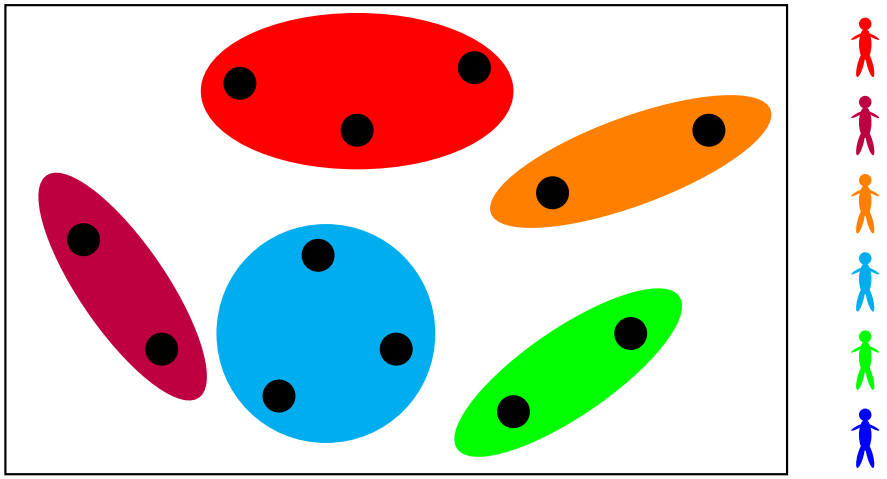
We already knew: [Feige,Mirrokni,V. '07]

- $(\frac{1}{2} + \epsilon)$-approximation for unconstrained submodular maximization in the *value oracle model* would need exponentially many queries

Note: For Max Cut, $\frac{1}{2}$ can be improved using SDP.
Could there be $\frac{1}{2} + \epsilon$ for every succinctly represented special case?
(UG-hardness of 0.7-approximation for a special case [Austrin '10])

Submodular welfare maximization

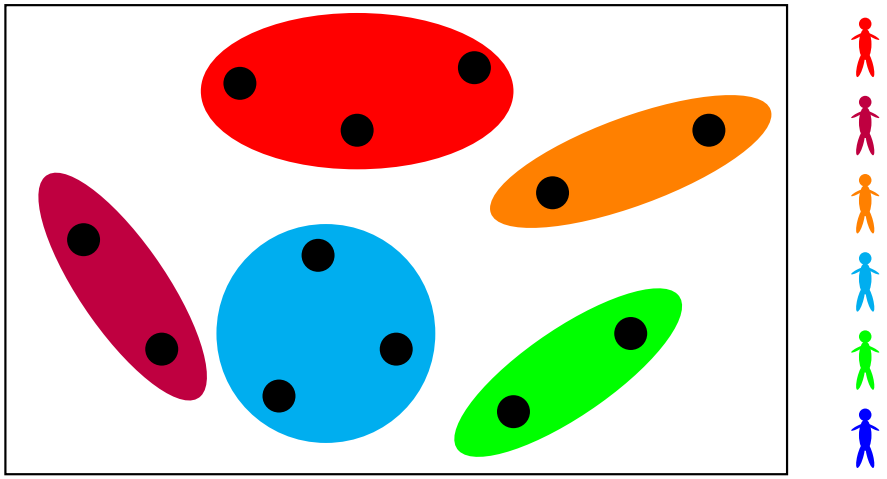
Given: m items, n players with *submodular valuations* $v_i : 2^{[m]} \to \mathbb{R}_+$.
Goal: Find an allocation of disjoint sets $S_1, \ldots, S_n$ to the n players, maximizing the *social welfare* $\sum_{i=1}^{n} v_i(S_i)$.



Allocation $(S_1, S_2, \ldots, S_n)$ has value $\sum_{i=1}^{n} v_i(S_i)$.

Submodular welfare maximization

Given: m items, n players with *submodular valuations* $v_i : 2^{[m]} \to \mathbb{R}_+$.
Goal: Find an allocation of disjoint sets $S_1, \ldots, S_n$ to the n players, maximizing the *social welfare* $\sum_{i=1}^{n} v_i(S_i)$.



Allocation $(S_1, S_2, \ldots, S_n)$ has value $\sum_{i=1}^{n} v_i(S_i)$.

- $(1 - 1/e)$-approximation [V. '08], optimal by [Khot,Lipton,Markakis,Mehta '05]
- $(1 - (1 - 1/n)^n)$-approximation for n players [Feldman,Naor,Schwartz '11], optimal for fixed n in the value oracle model [Mirrokni,Schapira,V. '08]

What about succinctly represented valuations?

Our results

Oracle hardness $\Rightarrow$ Computational hardness
for submodular optimization problems:

- There are succinctly represented nonnegative submodular functions for which there is no $(\frac{1}{2} + \epsilon)$-approximation for the problem $Max_{S \subseteq X}\ f(S)$, unless $NP = RP$.

Our results

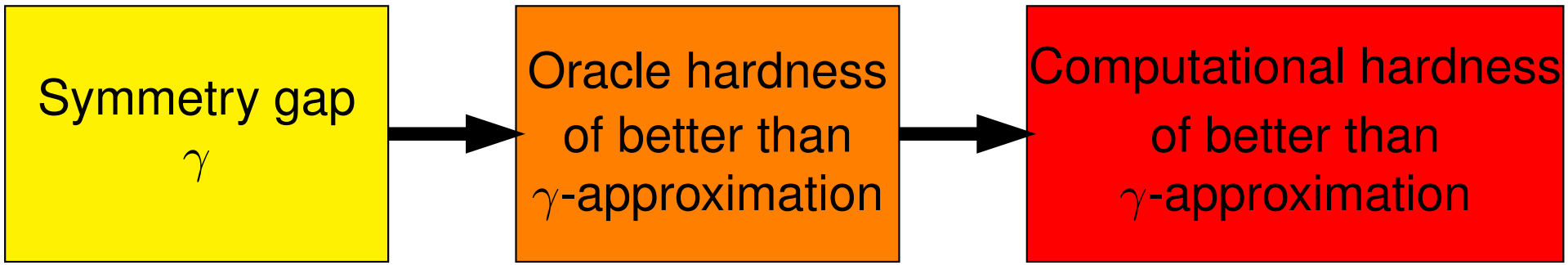
Oracle hardness $\Rightarrow$ Computational hardness
for submodular optimization problems:

- There are succinctly represented nonnegative submodular functions for which there is no $(\frac{1}{2} + \epsilon)$-approximation for the problem $Max_{S \subseteq X}\ f(S)$, unless $NP = RP$.
- There are succinctly represented monotone submodular functions for which there is no $(1 - (1 - \frac{1}{n})^n + \epsilon)$-approximation for welfare maximization with n players, unless $NP = RP$.

Our results

Oracle hardness $\Rightarrow$ Computational hardness
for submodular optimization problems:

- There are succinctly represented nonnegative submodular functions for which there is no $(\frac{1}{2} + \epsilon)$-approximation for the problem $Max_{S \subseteq X}\ f(S)$, unless $NP = RP$.
- There are succinctly represented monotone submodular functions for which there is no $(1 - (1 - \frac{1}{n})^n + \epsilon)$-approximation for welfare maximization with n players, unless $NP = RP$.
- More generally, any hardness result proved by the **symmetry gap** technique [V. '09] can be converted into computational hardness.



Hardness from symmetry gap

Intuitive statement: *For any instance $\mathcal{I}$ where the gap between the best symmetric and best asymmetric solution is γ, there is no approximation better than γ for instances "similar to $\mathcal{I}$".*

Applications: Monotone submodular maximization

Constraint	Approximation	Hardness	hardness ref.
$\lvert S\rvert \leq k$, matroid	$1-1/e$	$1-1/e$	[Nemhauser,Wolsey '78] [Feige '98]
n-players welfare	$1-(1-\frac{1}{n})^n$	$1-(1-\frac{1}{n})^n$	[Mirrokni,Schapira,V. '08]

Non-monotone submodular maximization

Constraint	Approximation	Hardness	hardness ref.
unconstrained	$1/2$	$1/2$	[Feige,Mirrokni,V. '07]
$\lvert S\rvert \leq k$	$1/e$	0.49	[Oveis-Gharan,V. '11]
matroid	$1/e$	0.48	[Oveis-Gharan,V. '11]
matroid base	$\frac{1}{2}(1-\frac{1}{\nu})$	$1-\frac{1}{\nu}$	[V. '09]

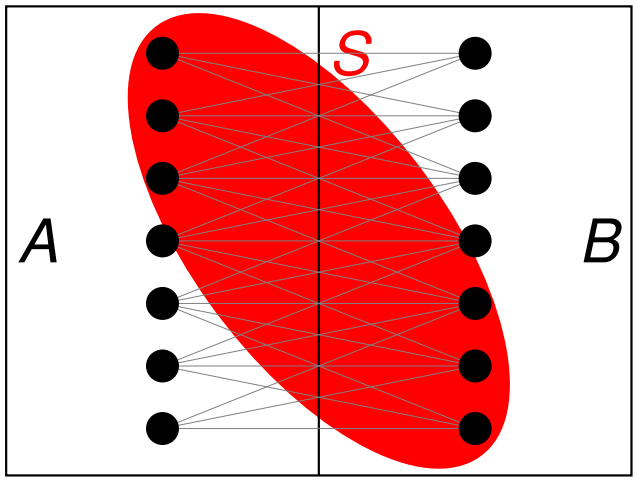
Our techniques

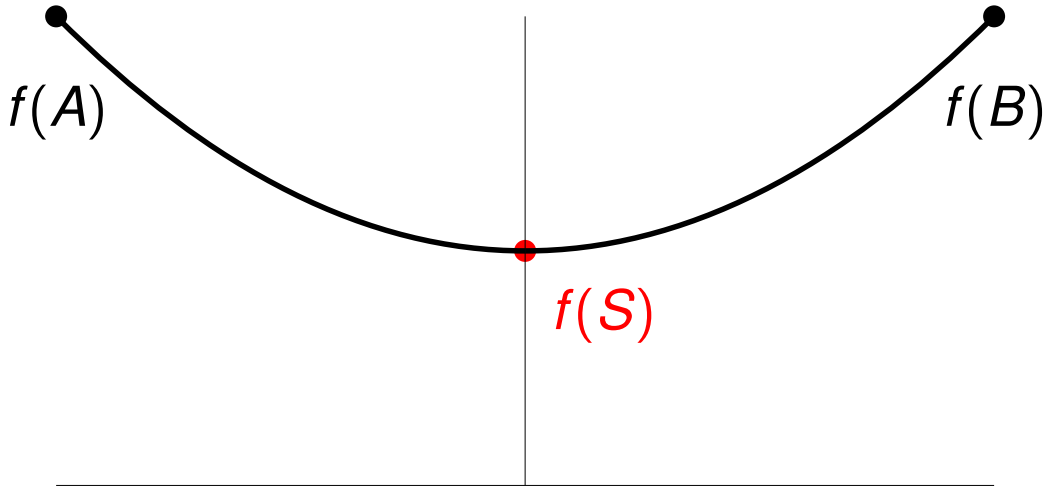
How do we convert oracle hardness into computational hardness?

- We use objective functions of the **same type** that were used in the oracle hardness proofs.
- We do not use the PCP theorem or Unique Games Conjecture.
- We encode the optimal set **implicitly** as a solution to a computationally difficult problem.
- To evaluate the function efficiently, we use **list decodable codes**.

Hardness of unconstrained submodular maximization

Oracle hardness proof [Feige,Mirrokni,V. '07]
starts from the cut function of a complete bipartite graph:



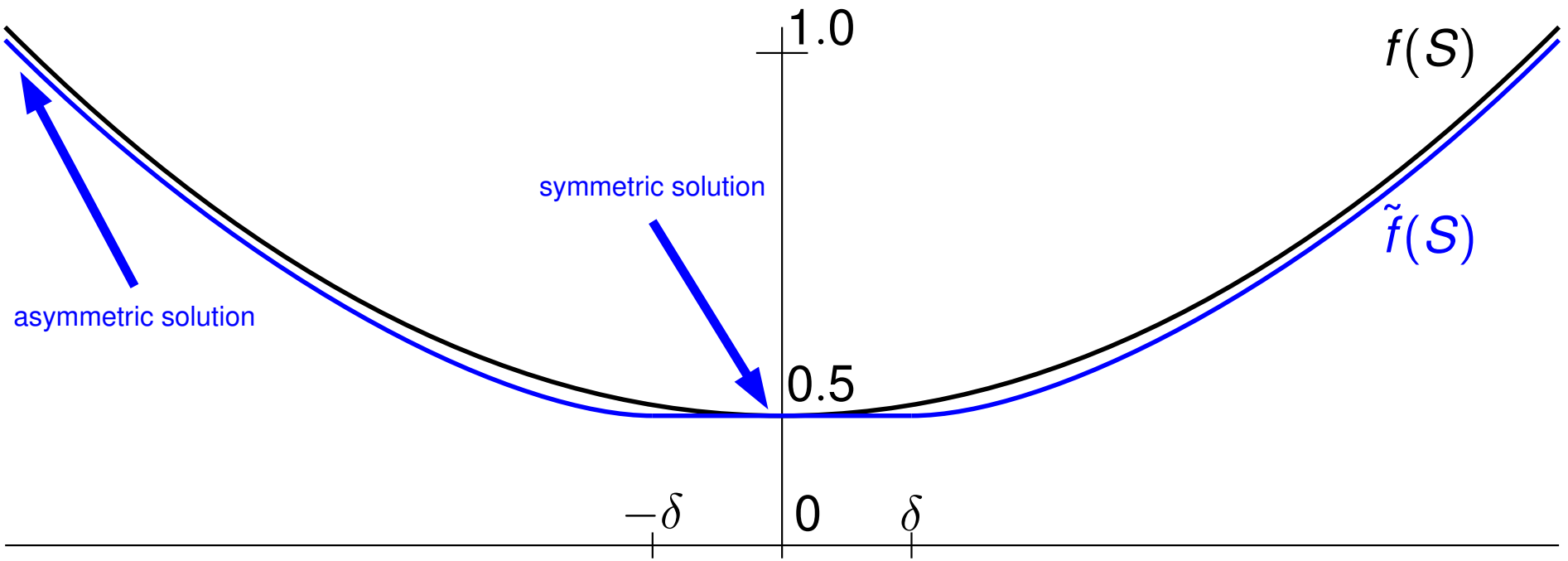


- $f(S) = \psi(x, y) = x(1-y) + (1-x)y$, where $x = \frac{|S \cap A|}{|A|}, y = \frac{|S \cap B|}{|B|}$
- $f(A) = f(B) = 1$
- $f(S) = \frac{1}{2}$ if $|S \cap A| = |S \cap B| = \frac{1}{2}|A| = \frac{1}{2}|B|$

The oracle hardness argument

For a suitable perturbation of $f(S)$, the partition (A, B) cannot be found using poly-many value queries

$\Rightarrow$ only symmetric solutions can be found efficiently.



$\Rightarrow$ max $f(S)$ cannot be solved better than within 0.5.

How to represent the functions succinctly?

What is the issue: imagine we want to present the function

$$f(S) = \psi(x, y) = x(1 - y) + (1 - x)y$$

where $x = \frac{|S \cap A|}{|A|}, y = \frac{|S \cap B|}{|B|}$, explicitly on the input.
We can encode it by writing down the formula and specifying (A, B).
However, then it's easy to determine the optimal solution A (or B)!

How to represent the functions succinctly?

What is the issue: imagine we want to present the function

$$f(S) = \psi(x, y) = x(1 - y) + (1 - x)y$$

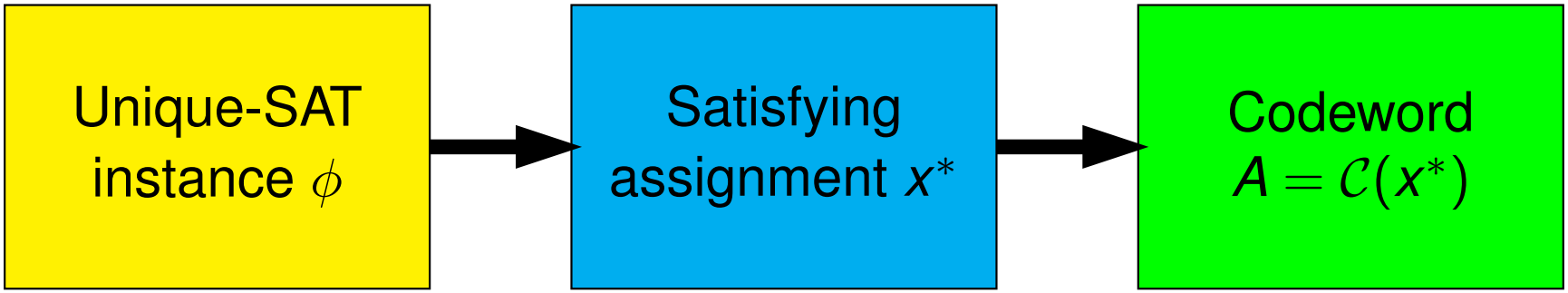
where $x = \frac{|S \cap A|}{|A|}, y = \frac{|S \cap B|}{|B|}$, explicitly on the input.
We can encode it by writing down the formula and specifying (A, B).
However, then it's easy to determine the optimal solution A (or B)!

Our approach:

- (A, B) is determined by the solution of a computationally difficult problem (i.e. SAT).
- Rather than (A, B), the encoding contains a SAT instance ϕ.
- *HOWEVER: this encoding must allow us to evaluate the function!*

Encoding by list-decodable codes

Encoding:
We encode the objective function by specifying a Unique-SAT formula ϕ on t variables, and a *list-decodable code* $\mathcal{C} : \{0,1\}^t \to \{0,1\}^n$.



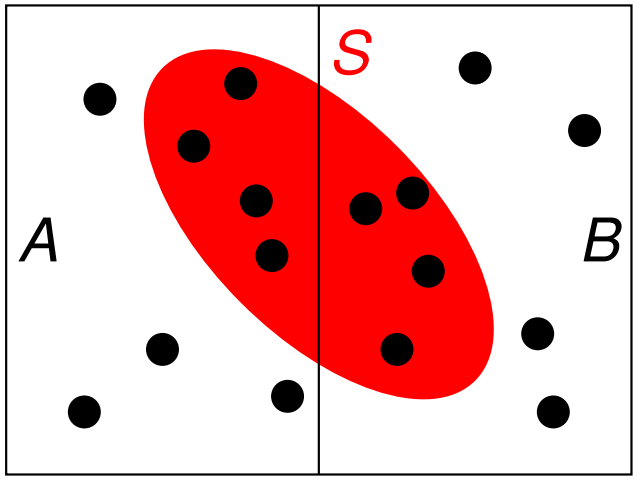
Interpretation: $(\mathcal{C}, \phi)$ encodes the *perturbed function* $\tilde{f}(S)$

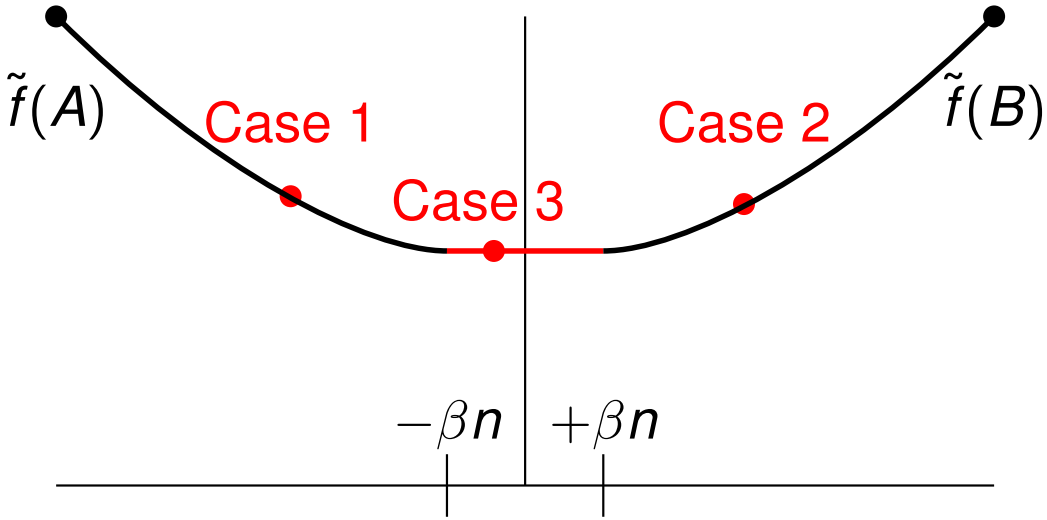
$$\tilde{f}(S) = \tilde{\psi}\left(\frac{|S \cap A|}{|A|}, \frac{|S \cap B|}{|B|}\right)$$

where $A = \overline{B} = \mathcal{C}(x^*)$, and $x^* =$ unique satisfying assignment to ϕ.

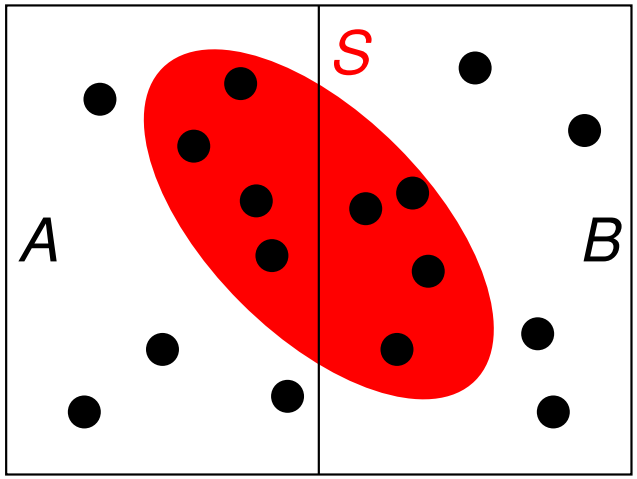
LDCs from [Guruswami-Rudra '08], Unique-SAT hardness from [Valiant-Vazirani '86]

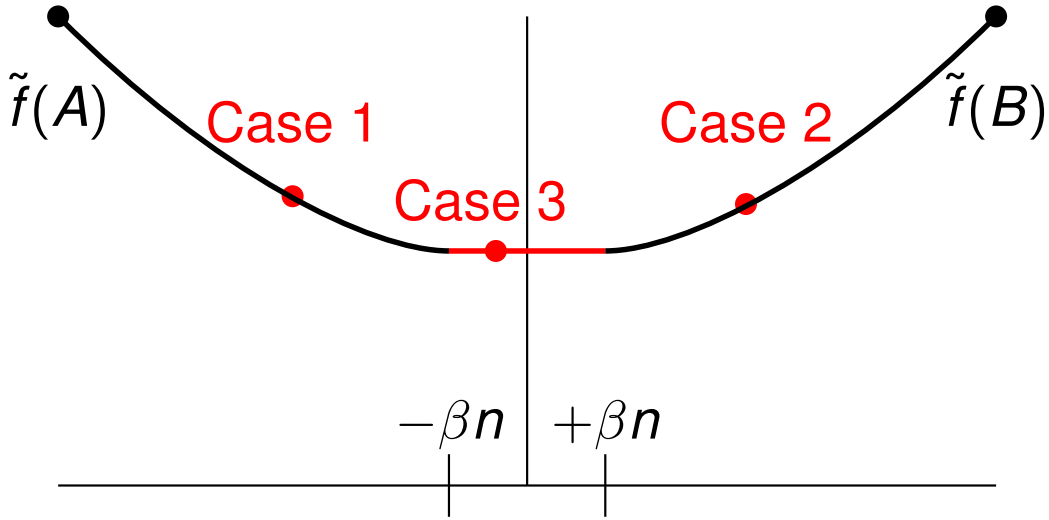
Evaluating $\tilde{f}(S)$ using this representation





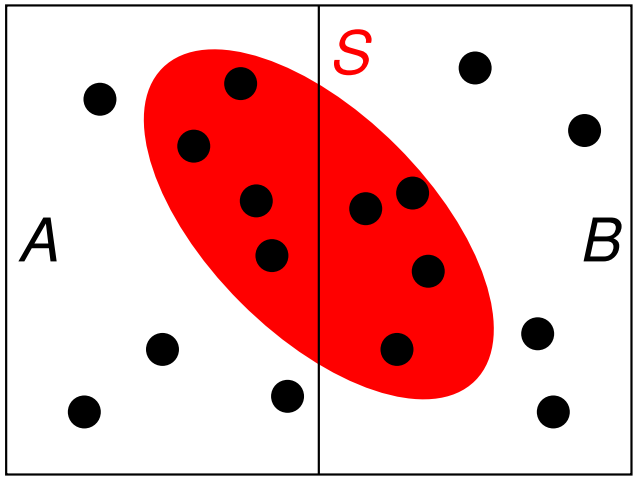
Evaluating $\tilde{f}(S)$ using this representation

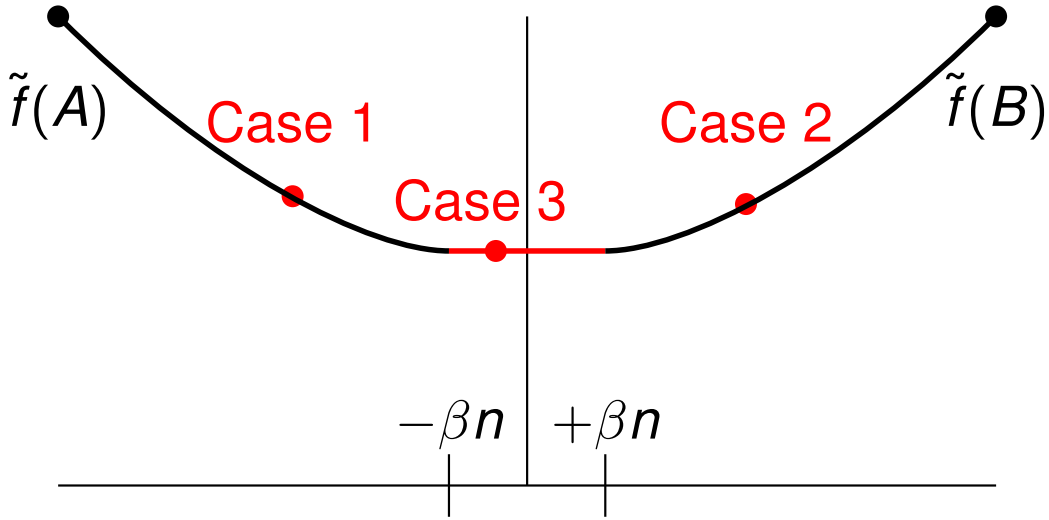




- Case 1: If $|S \cap A| - |S \cap B| > \beta n$, we find $x^* = C^{-1}(A)$ as one of the codewords obtained by list-decoding S. Evaluating $\phi(x^*)$ confirms that A is the correct set.

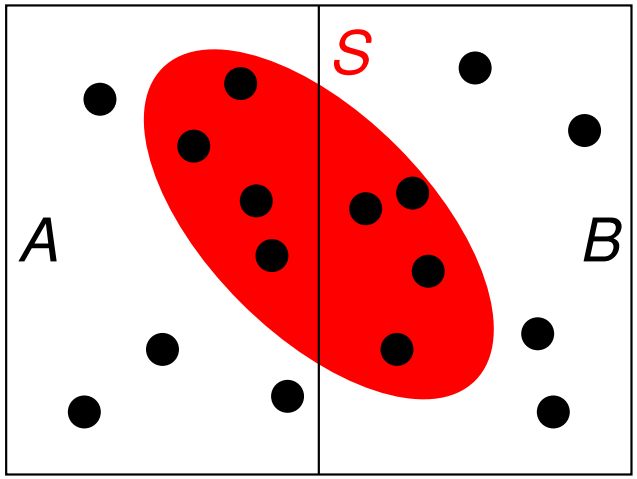
Evaluating $\tilde{f}(S)$ using this representation

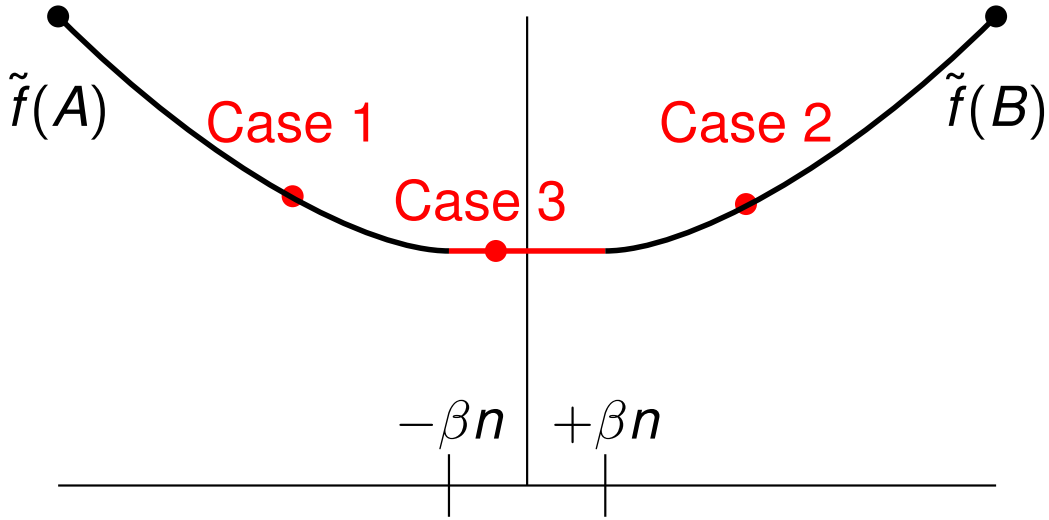




- Case 1: If $|S \cap A| - |S \cap B| > \beta n$, we find $x^* = \mathcal{C}^{-1}(A)$ as one of the codewords obtained by list-decoding S. Evaluating $\phi(x^*)$ confirms that A is the correct set.
- Case 2: If $|S \cap B| - |S \cap A| > \beta n$, we find $x^* = \mathcal{C}^{-1}(A)$ again by list-decoding $\overline{S}$.

Evaluating $\tilde{f}(S)$ using this representation





- Case 1: If $|S \cap A| - |S \cap B| > \beta n$, we find $x^* = \mathcal{C}^{-1}(A)$ as one of the codewords obtained by list-decoding S. Evaluating $\phi(x^*)$ confirms that A is the correct set.
- Case 2: If $|S \cap B| - |S \cap A| > \beta n$, we find $x^* = \mathcal{C}^{-1}(A)$ again by list-decoding $\overline{S}$.
- Case 3: If $|S \cap A| - |S \cap B| \in [-\beta n, +\beta n]$, we are not able to determine A and B. But $\tilde{f}(S)$ in this case depends only on $|S|$, so we can still evaluate $\tilde{f}(S)$.

Conclusions

We match several inapproximability results in the oracle model with computational hardness results.

Another application: Computational hardness result ruling out *truthful-in-expectation mechanisms* for combinatorial auctions.

[upcoming EC '12]

Conclusions

We match several inapproximability results in the oracle model with computational hardness results.

Another application: Computational hardness result ruling out *truthful-in-expectation mechanisms* for combinatorial auctions.

[upcoming EC '12]

Future questions:

- Encoding is not very natural - is there a hardness result for more natural objective functions?
- Is there a matching hardness result for objective functions in "Max-CSP form"? (sum over small gadgets)
- Does this technique apply to problems other than submodular optimization?