Combinatorial Optimization using Comparison Oracles

David Woodruff

CMU / Google

Joint work with:

Vincent Cohen-Addad, Anupam Gupta, Guru Guruganesh, Euiwoong Lee, Tommaso d'Orsi, Debmalya Panigrahi, Madhusudhan Pittu, Renato Paes Leme, Jon Schneider

Outline

- Model
- Unweighted cuts
- Weighted cuts / general results

Combinatorial Optimization

Given: $\mathcal{F} \subseteq 2^U$ is a family of *feasible subsets* over ground set U and $f: 2^U \to \mathbb{R}$ is a weight/cost function.

$$\text{solve } \max\{f(S): S \in \mathcal{F}\}, \min\{f(S): S \in \mathcal{F}\},$$

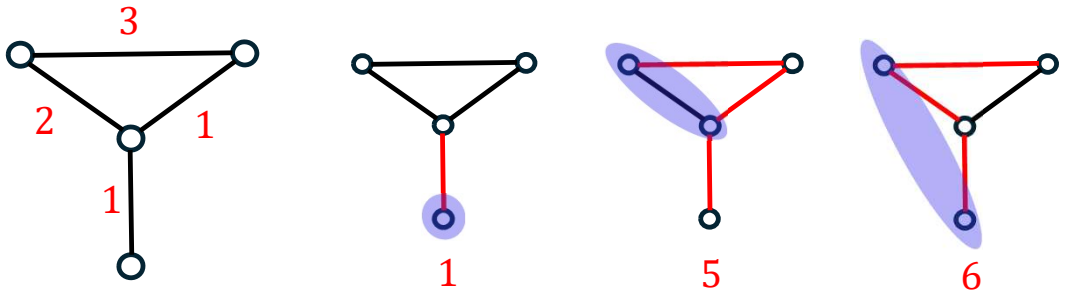
f is modular in most settings i.e., $f(S) = \sum_{e \in S} w_e$

Cut Optimization

A cut S $\subseteq V$ defines an edge indicator vector $e(S) \in \{0,1\}^{\frac{n(n-1)}{2}}$ of pairs {u,v} with $|S \cap \{u,v\}| = 1$

Then $f(S) =< w, e(S) >$ is modular over the family $\mathcal{F}$ = {e(S) : S $\subseteq \{0,1\}^n, S \neq \emptyset$ and $S \neq V$}

Note: Ground set U = $\{0,1\}^{\frac{n(n-1)}{2}}$ for cuts



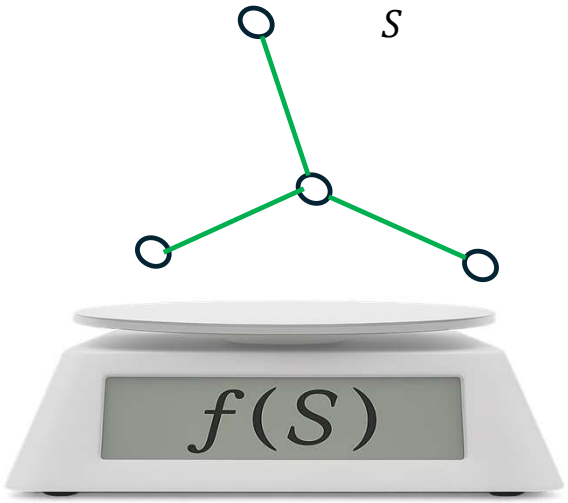
$G = (V, E)$ min cut max cut

Combinatorial Optimization

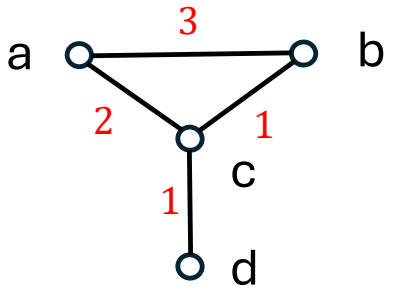
Given: $\mathcal{F} \subseteq 2^U$ is a family of *feasible subsets* over ground set U and $f: 2^U \to \mathbb{R}$ is a weight/cost function.

solve $\max\{f(S): S \in \mathcal{F}\}$, $\min\{f(S): S \in \mathcal{F}\}$,

Sometimes you have an explicit representation of $\mathcal{F}$ or a value oracle:



value oracle



f({a}) = 5

Comparison-Oracle Model

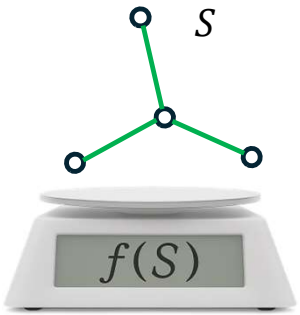
Given: $\mathcal{F} \subseteq 2^U$ is a family of *feasible subsets* over ground set U and $f: 2^U \rightarrow \mathbb{R}$ is a weight/cost function.

solve $\max\{f(S): S \in \mathcal{F}\}$, $\min\{f(S): S \in \mathcal{F}\}$,

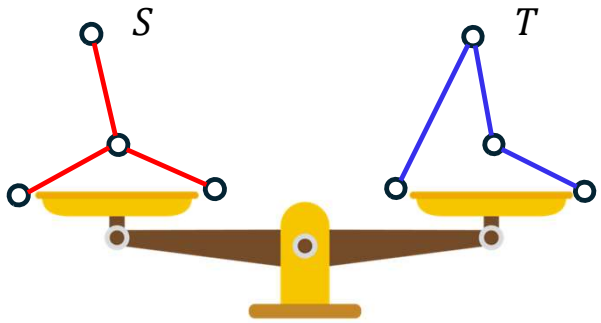
~~Sometimes you have an explicit representation of $\mathcal{F}$ or a value oracle~~:

Only allowed comparisons between $f(S)$ and $f(T)$ for $S, T \in \mathcal{F}$, so get sign(f(S) – f(T))

Learn f(S) > f(T) or
f(S) = f(T) or
f(S) < f(T))



value oracle



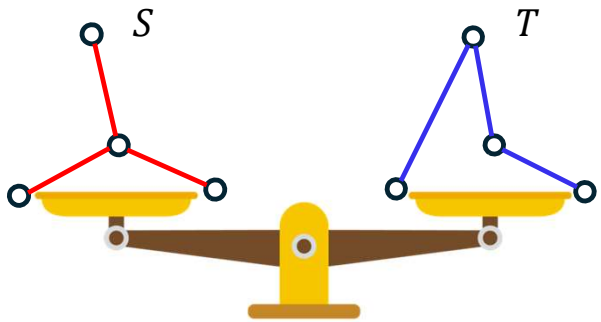
Comparison oracle

The Main Question

Can we find the **optimal feasible set** $S^* = \arg\min_{S \in \mathcal{F}} f(S)$ by comparing only $\mathrm{poly}(|U|)$ feasible sets?

Answer: Query-complexity answer: Yes for any family $\mathcal{F}$ and modular function f.

$O(n \log^2 n \log |\mathcal{F}|)$ queries! (recall: |U| = n)

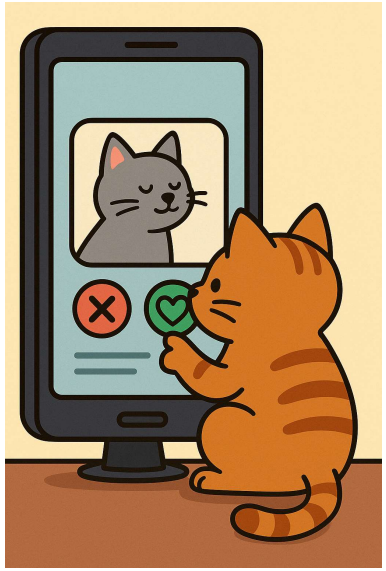


Comparison oracle

Motivation

Precise utilities often unavailable

In recommendation systems, fair division, or crowdsourcing, users rarely report explicit scores.
Only relative preferences.

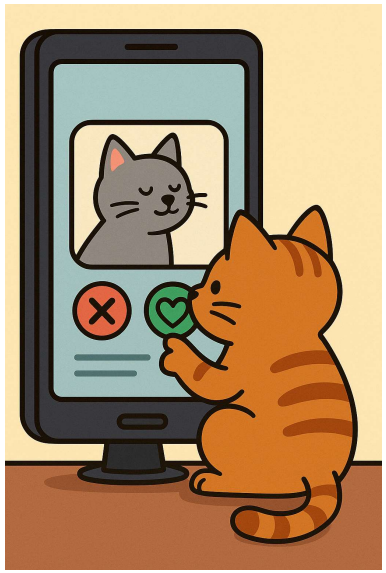


Recommendation systems

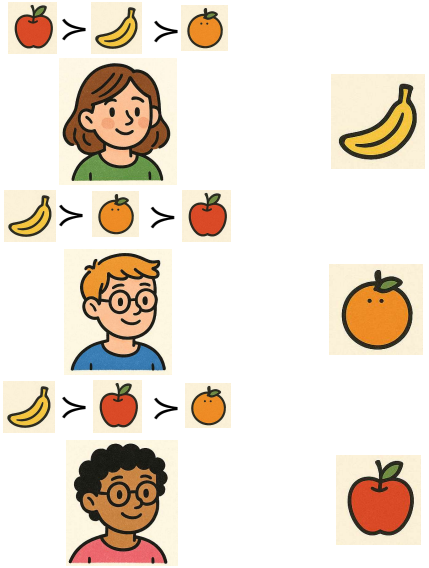
Motivation

Ordinal feedback is natural and robust

Easier to compare alternatives than assign numeric values; comparisons are stable under monotonic transformations.



Recommendation systems

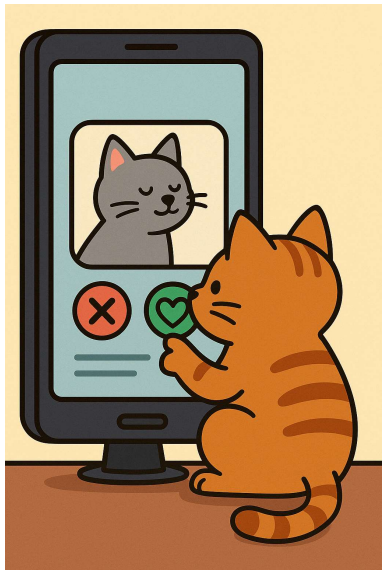


Fair division

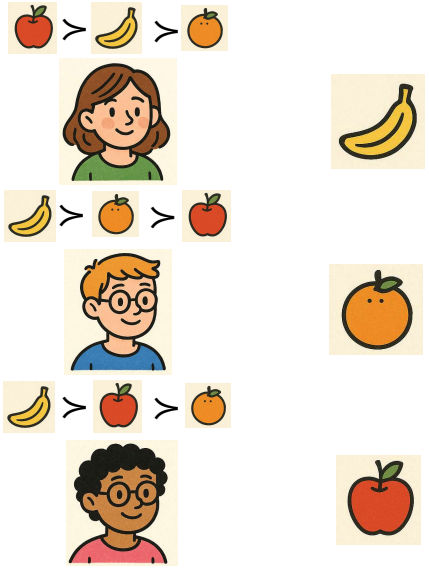
Motivation

Broader applicability under limited information

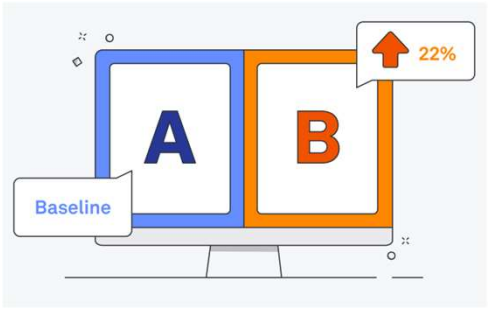
Enables optimization with minimal input: ideal for human-in-the-loop systems, black-box feedback, or uncertain environments.

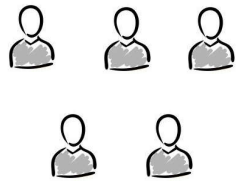


Recommendation systems



Fair division





Crowdsourcing

Our Results

1. Can solve unweighted mincut with $O\sim(n)$ queries and $O\sim(n^2)$ time

2. Can solve weighted mincut with $O\sim(n^3)$ queries and exp(n) time

3. Can optimize any modular function on any family $\mathcal{F}$ of subsets over a ground set {1, 2, …, n} with poly(n) queries

Unweighted Cuts

Theorem (Graph recovery): A simple unweighted graph G with $n \geq 4$ can be recovered using $\widetilde{O}(m)$ cut comparison queries in $\widetilde{O}(n^2)$ time.

Theorem (Minimum cut): There is a randomized algorithm that computes the minimum cut of a simple graph G using $\widetilde{O}(n)$ queries and $\widetilde{O}(n^2)$ time.

Cut query: compare cut values $|\delta(S)|$ and $|\delta(T)|$ for two cuts S and T

Unweighted Cuts

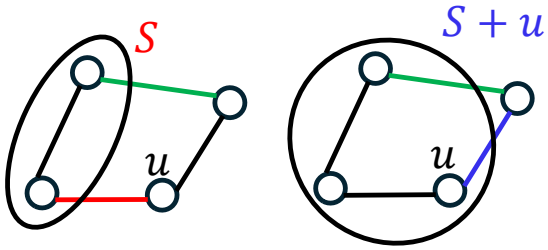
Theorem (Graph recovery): A simple unweighted graph G with $\mathrm{n} \geq 4$ can be recovered using $\widetilde{\mathrm{O}}(\mathrm{n}^2)$ cut comparison queries in $\widetilde{\mathrm{O}}(\mathrm{n}^2)$ time.

Idea: Reconstruct the neighborhood of each vertex using $\widetilde{\mathrm{O}}(n)$ queries.

Observation: For any $S \subseteq V$, $u \notin S$,

1. We can compare $|\delta(u,S)|$ and $|\delta(u,V\backslash S)|$ using one query.

Proof:
$$|\delta(S)| - |\delta(S+u)| = |\delta(u,S)| - |\delta(u,V\backslash S)|$$
$$= 2|\delta(u,S)| - |\delta(u)|$$



2. $|\delta(u,S)|$ is monotonic in S.

Unweighted Cuts

Theorem (Graph recovery): A simple unweighted graph G with $\mathrm{n} \geq 4$ can be recovered using $\mathrm{O}(\mathrm{n}^2)$ cut comparison queries in $\widetilde{\mathrm{O}}(\mathrm{n}^2)$ time.

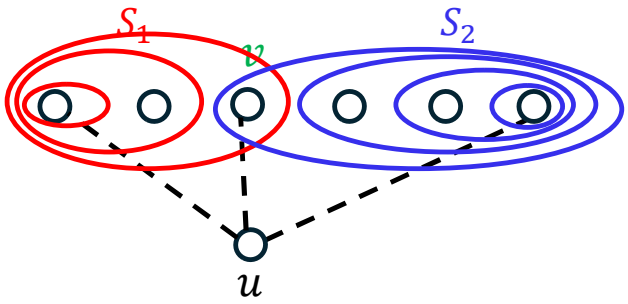
Idea: Reconstruct the neighborhood of each vertex using $O(n)$ queries.

Can improve to $\widetilde{\mathrm{O}}(\delta(\mathrm{u}))$

Observation: For any $S \subseteq V$, $u \notin S$,

1. We can compare $|\delta(u,S)|$ and $|\delta(u,V\backslash S)|$ using one query.

2. $|\delta(u,S)|$ is monotonic in S.



Algorithm

1. Identify S_1 and S_2 such that
   $|\delta(u,S_i)| = \left\lfloor \frac{1}{2}|\delta(u)| \right\rfloor$ and $S_1 \cap S_2 = \emptyset$. ⟶ $\mathrm{O}(\log \mathrm{n})$ queries

2. For any $v \notin S_i$, $\{u,v\} \in E$ iff ⟶ 1 query
   $$|\delta(u, S_i + v)| > \frac{1}{2}|\delta(u)|$$

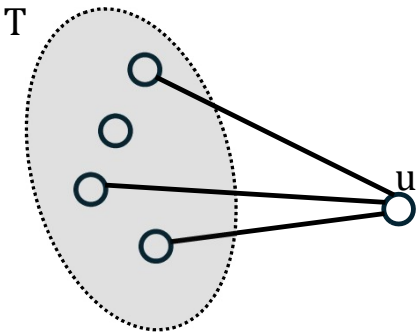
Unweighted Cuts

Theorem (Minimum cut): There is a randomized algorithm that computes the minimum cut of a simple graph G using $\widetilde{O}(n)$ queries and $\widetilde{O}(n^2)$ time.

Main idea: Implement the Rubenstein, Schramm, Weinberg algorithm using cut comparisons.

Structural primitive: For any vertex u and subset T with $u \notin T$ extract r edges from $\delta(u, T)$ using $\widetilde{O}(r)$ queries and $\widetilde{O}(n + |T|)$ time .

Proof using the algorithm from previous slide.



This structural primitive can be used to sample edges of a contracted graph G' independently with any probability p using $|E'| \cdot O\sim(p)$ + O~(n) queries.

This is the main primitive in the RSW algorithm.

This gives an $\widetilde{O}(n)$ query algorithm for finding the minimum cut.

Weighted Cuts

Theorem: For family $\mathcal{F} \subseteq 2^U$ and unknown $w: U \to \mathbb{R}$ with $|U| = n$

can solve $\min_{S \in \mathcal{F}} w(S)$ using $O\left(n \log^2 n \cdot \log|\mathcal{F}|\right) = \tilde{O}(n^2)$ queries!

Theorem (Global Subspace Learning): If $w \in \{-B, \ldots, 0, \ldots, B\}^n$, we can in fact sort $\{w(S): S \in \mathcal{F}\}$ using $O(n\mathrm{B} \log n\mathrm{B})$ comparison queries.

First result uses inference dimension result of Kane, Lovett, Moran and Zhang

General Boolean Linear Optimization Results

Corollary: An $\tilde{O}(n^3)$ cut query algorithm to find the min/max cut in any graph with unknown arbitrary weights (can be negative).

Corollary: An $\tilde{O}(n^2\mathrm{B})$ cut query algorithm to find the min/max cut in any graph with integral weights bounded by B.

$O(\exp(n))$ time

Lemma: For a planar graph G with unknown edge weights w: $E \to \{0, 1, .., B\}$, the min/max cuts can be found using $O(n\mathrm{B} \log n\mathrm{B})$ comparison queries and poly(n, B) time.

→ Efficient implementation of GSL

GSL via Affine Hulls: Warmup

Theorem: If $w \in \{0,1\}^n$, we can in fact sort $\{w(S): S \in \mathcal{F}\}$ using $O(n \log n)$ comparison queries. → $O(n^2 \log n)$

Execute insertion sort. Only $n+1$ possible values for $w(S)$.

Sets seen so far organized into buckets $B_0, B_2, \dots, B_n$ which are ordered by weight.

Maintain $\text{AffineHull}(\{\mathbb{1}_T: T \in B_i\})$ for each bucket.

$\text{AffineHull}(x_1, \dots, x_k) = x_1 + \text{span}(x_2 - x_1, \dots, x_k - x_1)$

If $x \in \text{AffineHull}(x_1, \dots, x_k)$, then $f(x) = f(x_1) + \sum_j f(x_j - x_1) = f(x_1)$

INFER-or-INSERT:

When new set S arrives,

1. infer S is contained in B_i if $\mathbb{1}_S \in \text{AffineHull}(\{\mathbb{1}_T: T \in B_i\})$ for some $0 \le i \le n$.
2. insert S (possibly creating a new bucket) via binary search using $O(\log t)$ comparisons.

We only pay for inserts. There can be at most $O(n^2)$ inserts.

1. Inserts that create new buckets are at most $n+1$
2. When S is inserted into existing bucket B_i, the dimension of $\text{affine\_hull}(\{\mathbb{1}_T: T \in B_i\})$ increases. This happens at most $O(n^2)$ times (n times for each bucket).

GSL via Affine Hulls (Improved)

Theorem: If $w \in \{0,1\}^n$, we can in fact sort $\{w(S): S \in \mathcal{F}\}$ using $O(n \log n)$ comparison queries.

Execute insertion sort. Only n possible values for $w(S)$.

Sets seen so far organized into buckets $B_1, B_2, \ldots, B_t$ which are ordered by weight.

~~Maintain AffineHull($\{\mathbb{I}_T : T \in B_i\}$) for each bucket.~~

Maintain $\mathcal{A} = \text{span}(\cup_i \{\mathbb{I}_S - \mathbb{I}_T : S, T \in B_i\})$ – "learned" subspace orthogonal to w.

Now: S is contained in B_i if $\mathbb{I}_S = \mathbb{I}_T + v$, for some $v \in \mathcal{A}$, where $f(\mathbb{I}_T) = i$ and $\mathbb{I}_T$ is the bucket representative for B_i

INFER-or-INSERT:
When new set S arrives,

1. infer S is contained in B_i for some $0 \leq i \leq n$.
2. insert S (possibly creating a new bucket) via binary search using $O(\log t)$ comparisons.

Again, only pay for inserts. There can be at most O(n) inserts.

Open Problems

1. Minimum cut in weighted graphs using polynomial time and queries?

2. For what general class of feasible sets $\mathcal{F}$ can we get polynomial time algorithms?

3. Our general algorithms also solve max-cut, so we need some assumptions on $\mathcal{F}$, but maybe:

   Can we do all of submodular minimization with poly(n) queries using a comparison oracle?