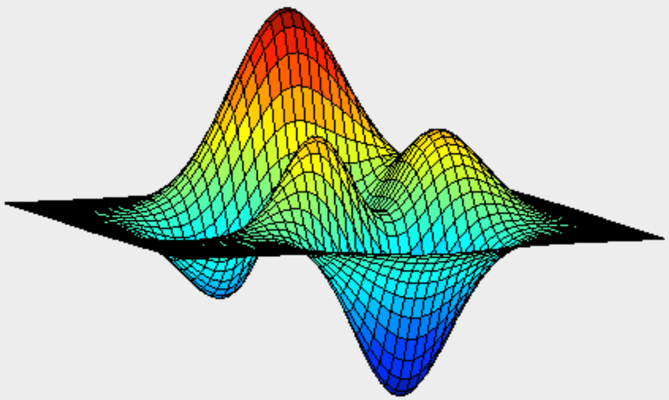
Submodular Maximization

Niv Buchbinder

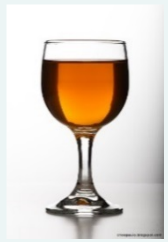
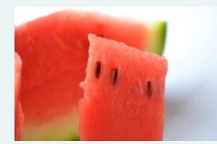
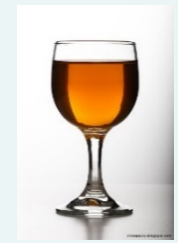
Tel-Aviv university

Goals of talk

- What are submodular functions and why are they interesting?
- Classical approaches for maximizing submodular functions.
- New(er) approaches based on relaxations.
- Open problems and research directions.



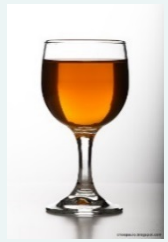
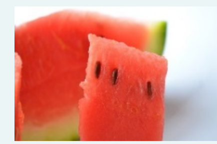
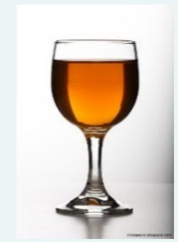
Example 1: Adding a Dessert

Meal 1	Meal 2
  	    

- What is the added value of a dessert to each meal?

  Answer: dessert is worth **more** in meal 1

Example 1: Adding Dessert

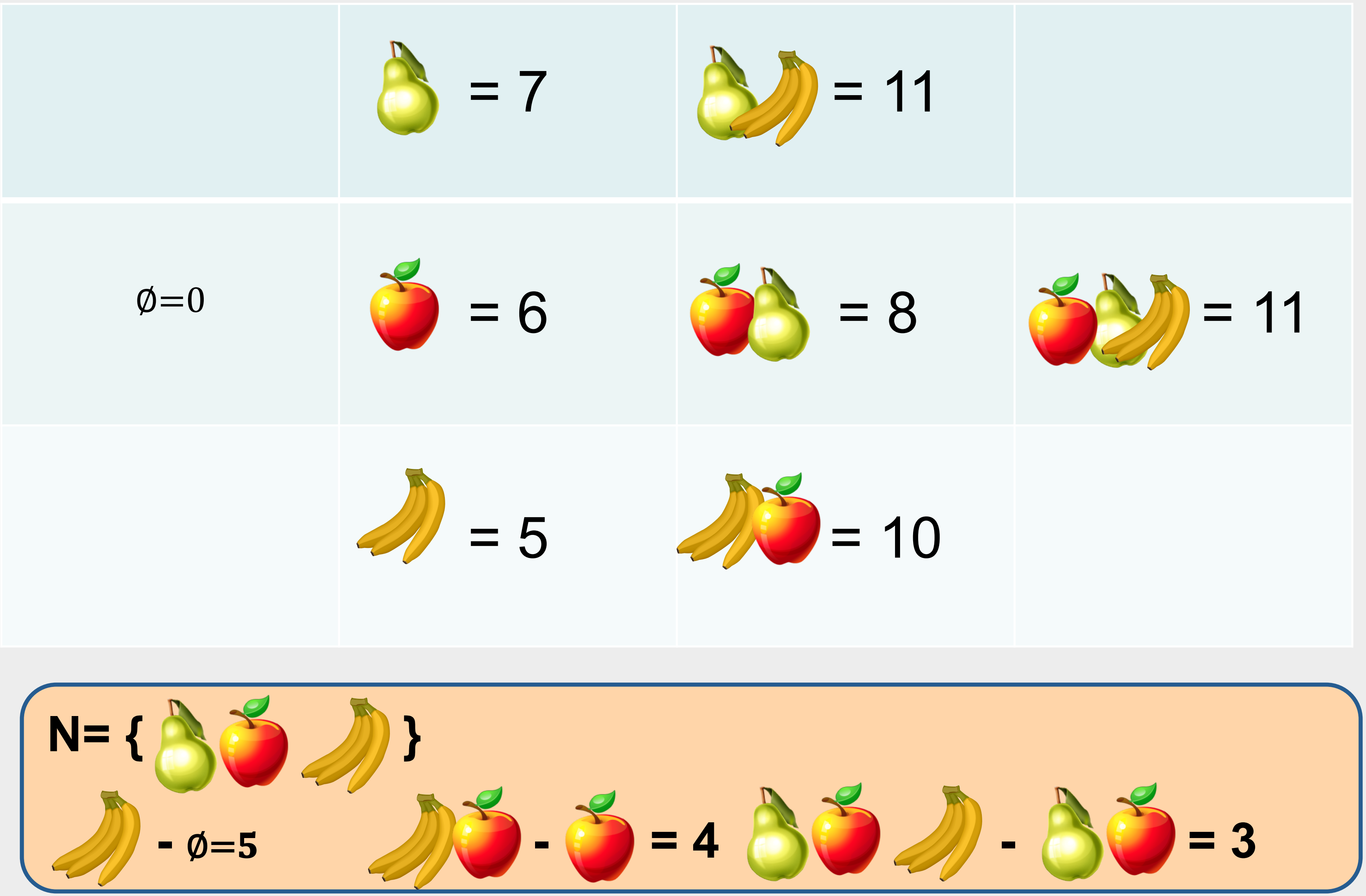
Meal 1	Meal 2
  	    

- $N = \{1, 2, \ldots, n\}$.

$f: 2^N \to \mathbb{R}_+$ is submodular if either (equivalent definitions):

- $f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B) \quad \forall A \subseteq B \subseteq N, u \notin B$
- $f(S) + f(T) \geq f(S \cup T) + f(S \cap T) \quad \forall S, T \subseteq N$

Example 2



Submodular Functions: More

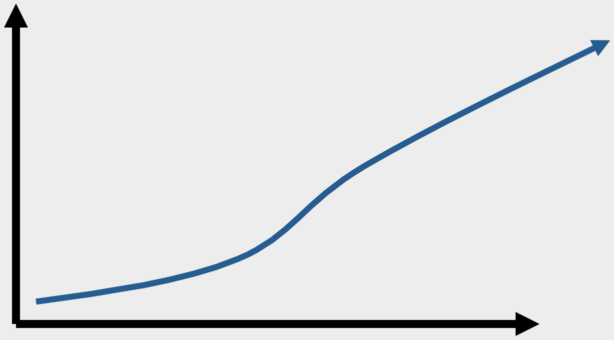
Representation of ‘f’ may be very large!

Value Oracle:

- Given a subset $S \subseteq N$, returns $f(S)$.

Monotonicity

- For every $A \subseteq B$, $f(A) \leq f(B)$ **“additional dessert never hurts”**



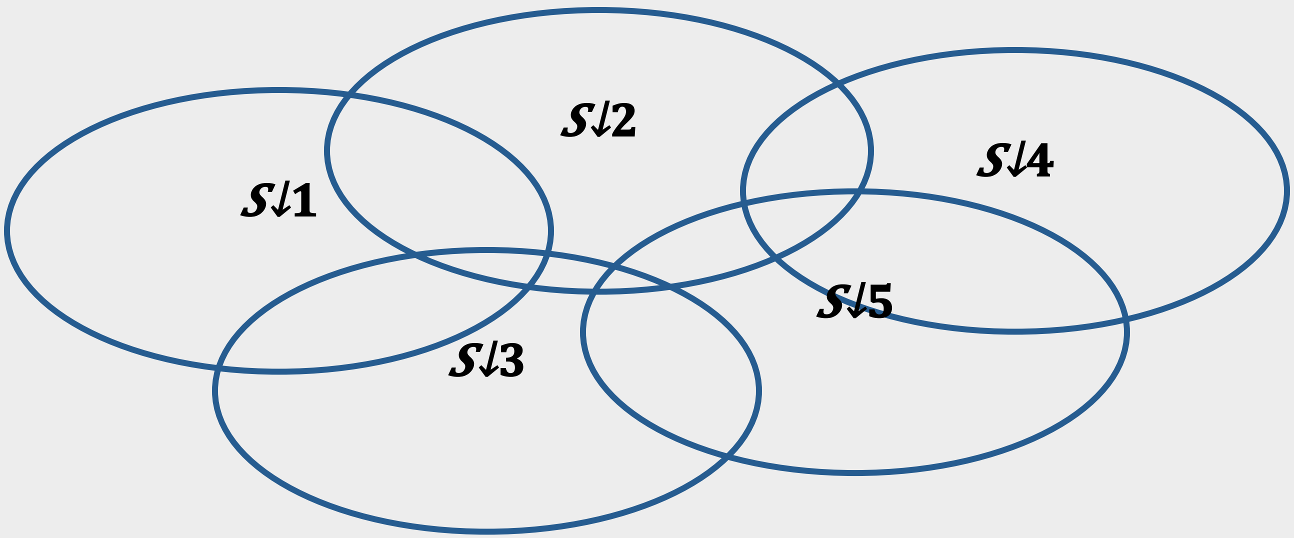
Coverage functions

Set cover:

- Elements $E = \{e_1, e_2, \ldots, e_n\}$
- Sets: $s_1, s_2, \ldots s_m \subseteq E$.

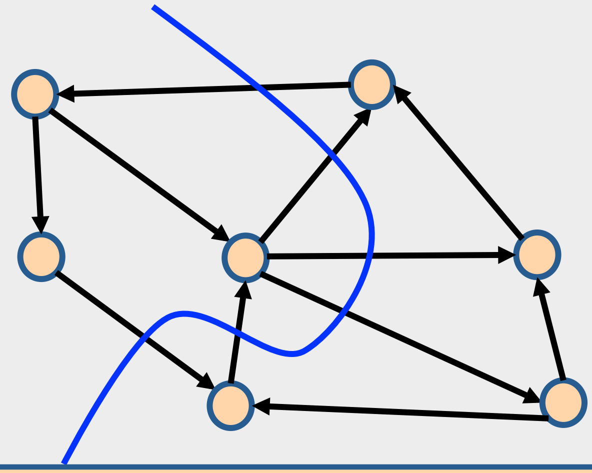
$\forall S = \{s_{i_1}, s_{i_2}, \ldots, s_{i_k}\}$ $\quad f(S) = |\cup_{s_i \in S} s_i|$

Observation: $f(\cdot)$ is a monotone submodular function.



Cut (capacity) functions

- (Directed) graph $G=(V,E)$ with capacities $c_e \geq 0$ on edges.
- $\forall S \subset V \quad f(S) = \sum_{u \in S,\, v \notin S} c_{u \to v}$
- **Observation (exercise):** $f(\cdot)$ is a (non-monotone) submodular function.



Additional motivation: Combinatorics, Algorithmic Game Theory, Learning …

Maximizing Submodular Functions

Unconstrained submodular function maximization

- $\max_{S \subseteq N} \{f(S)\}$: Find the best meal (only interesting if non-monotone)
- Generalizes Max (directed) cut.

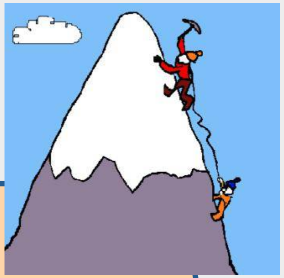
Submodular maximization with a cardinality constraint

- $\max_{S \subseteq N, |S| \leq k} \{f(S)\}$: Find the best meal of at most k dishes.
- Generalizes Max-k-coverage, Max cut with specified size.

Other constraints:

- Exactly k elements, Matroid, Packing …

Cardinality Constraints $\leq k$



Greedy Algorithm:

Start with the empty solution ($S_0 = \emptyset$)

For $i = 1, 2, \ldots k$:

- Add to solution the element contributing **the most**:

$$u = \arg\max_{u \in N} \{ f(S_{i-1} \cup \{u\}) - f(S_{i-1}) \},$$

$$S_i = S_{i-1} \cup \{u\}$$

Theorem [Nemhauser et al. 78]: Greedy is $(1 - 1/e)$-approximation for maximizing a monotone submodular function.

Theorem [Nemhauser et al. 78]: This is best possible.

Analysis

$$f(S_i) - f(S_{i-1})$$

$$\geq \frac{1}{k}\left(\sum_{u \in OPT \setminus S_{i-1}} f(S_{i-1} \cup \{u\}) - f(S_{i-1})\right) \quad \text{(Greedy)}$$

$$\geq f(S_{i-1} \cup OPT) - f(S_{i-1}) / k \quad \text{(Submodularity)}$$

$$\geq f(OPT) - f(S_{i-1}) / k \quad \text{(Monotonicity)}$$

➔ $f(OPT) - f(S_i) \leq (1 - 1/k) \cdot (f(OPT) - f(S_{i-1}))$

➔ $f(OPT) - f(S_k) \leq (1 - 1/k)^k \cdot (f(OPT) - f(S_0))$

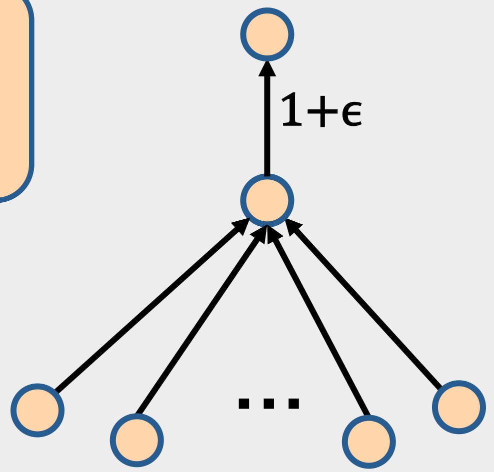
➔ $f(S_k) \geq (1 - 1/e) \cdot f(OPT)$

Problems with Greedy Approach

Unfortunately (fortunately), the greedy algorithm is not optimal for even slightly more general problems.

Partition matroid: $1/2$-approximation **[Fisher et al. `78]**
… can be improved (later)

Non-monotone submodular functions
… can be improved (later)



Remedy: Random Greedy

Random Greedy[B., Feldman, Naor, Schwartz '14]:

Start with an empty solution ($S_0 = \emptyset$)

For $i=1,2, \ldots k$:

- Add to the solution a **random** element among the k elements contributing the **most**.

Theorem **[B-Feldman-Naor-Schwartz14]:** Random greedy is:

- (1-$1/e$)-approximation for monotone submodular functions.
- ($1/e$)-approximation for (non-monotone) submodular functions.

Analysis (Monotone Functions)

$E[f(S_i)] - f(S_{i-1})$ — Fix event S_{i-1}

$\geq 1/k \left(\sum_{u \in OPT \setminus S_{i-1}} f(S_{i-1} \cup \{u\}) - f(S_{i-1}) \right)$ — (rand. Greedy)

$\geq f(S_{i-1} \cup OPT) - f(S_{i-1})/k$ — (Submodularity)

$\geq f(OPT) - f(S_{i-1})/k$ — (Monotonicity)

➔ Unfixing event S_{i-1}:

$f(OPT) - E[f(S_i)] \leq (1 - 1/k) \cdot (f(OPT) - E[f(S_{i-1})])$

➔ $f(OPT) - E[f(S_k)] \leq (1 - 1/k)^k \cdot (f(OPT) - f(S_0))$

➔ $E[f(S_k)] \geq (1 - 1/e) \cdot f(OPT)$

Analysis (Non-Monotone Function)

$E[f(S_i)] - f(S_{i-1})$ — Fix event S_{i-1}

$\geq \frac{1}{k} \left(\sum_{u \in OPT \setminus S_{i-1}} f(S_{i-1} \cup \{u\}) - f(S_{i-1}) \right)$ — (Greedy)

$\geq \left(f(S_{i-1} \cup OPT) - f(S_{i-1}) \right) / k$ — (Submodularity)

$\geq \left(f(OPT) - f(S_{i-1}) \right) / k$ — (Monotonicity)

Lemma: Let S be a random set such that $\Pr[u \in S] \leq p$. Then,

$E[f(S \cup OPT)] \geq (1-p) f(OPT)$

Unfixing event S_{i-1} : $\Pr[u \in S_i] \leq 1 - (1 - 1/k)^i$

➔ $E[f(S_i \cup OPT)] \geq (1 - 1/k)^i f(OPT)$

➔ Solving recursion we get $1/e$ approximation.

Analysis (Non-Monotone Function)

Lemma': Fix S_{i-1}. Then, $\mathrm{E}[f(S_i \cup OPT)] \geq (1-1/k) f(S_{i-1} \cup OPT)$

Rest of proof follows by unfixing S_{i-1} + induction

Proof: Let M_i be the set of k "top elements" at round i:

$\mathrm{E}[f(S_i \cup OPT)] - f(S_{i-1} \cup OPT)$

$= 1/k \sum_{u \in M_i} [f(S_{i-1} \cup \{u\} \cup OPT) - f(S_{i-1} \cup OPT)]$

$\geq^{(1)} 1/k \, (f(S_{i-1} \cup M_i \cup OPT) - f(S_{i-1} \cup OPT)) \geq^{(2)} -1/k \, f(S_{i-1} \cup OPT)$

(1): Submodularity (2): f is non-negative.

Random Greedy

Is randomization necessary?

Not for this case …

Theorem **[B., Feldman '16]:** Random greedy can be derandomized.

Combinatorial approach (greedy/local search) failed to improve bounds for most problems …

The Multilinear Relaxation

Relaxation in the linear case:

The Problem	Relaxation
$Max\ w \cdot x$ s.t: $x \in I \subseteq \{0,1\}^n$	$Max\ w \cdot x$ s.t: $x \in P \subseteq [0,1]^n$

- Objective function is the same for integral vectors.

Relaxation in the submodular case:

The Problem	Relaxation
$Max\ f(x)$ s.t: $x \in I \subseteq \{0,1\}^n$	$Max\ F(x)$ s.t: $x \in P \subseteq [0,1]^n$

Question: How to extend f beyond $\{0,1\}^n$?

The Multilinear Extension

Definition: For a vector $x \in [0,1]^N$:

$F(x)$= expected value of f on a random set containing each element e with probability x_e, **independently**.

$$F(x) = \sum_{S \subseteq N} f(S) \prod_{e \in S} x_e \prod_{e \notin S} (1 - x_e)$$

Observation: $F(x) = f(x)$ for integral vectors.

Properties of the Multilinear Extension

$$F(x) = \sum_{S \subseteq N} f(S) \prod_{e \in S} x_e \prod_{e \notin S} (1 - x_e)$$

$F(x)$ is not convex nor concave. But,

1. f is monotone ➔ $\partial F(x)/\partial x_e \geq 0$ / $f(S \cup \{e\}) \geq f(S)$
2. $\partial^2 F(x)/\partial x_e^2 = 0$ (F is multilinear …)
3. $\partial^2 F(x)/\partial x_i \partial x_j \leq 0$ / $f(S \cup \{e_j, e_i\}) - f(S \cup \{e_i\}) \leq f(S \cup \{e_j\}) - f(S)$
4. $F(x)$ is concave along directions $d \geq 0$.

Properties We Use

$$F(x)=\sum_{S\subseteq N} f(S)\prod_{e\in S} x_e \prod_{e\notin S}(1-x_e)$$

1. $\sum_{e\in A}[F(x\vee 1_e)-F(x)]\geq F(x\vee 1_A)-F(x)$

$(x\vee y)_e=\max\{x_e, y_e\}$

2. If f is monotone ➔ $F(x\vee 1_A)\geq F(x)$.

Observation: $\partial F(x)/\partial x_e = \left(F(x\vee 1_e)-F(x)\right)/(1-x_e)$

Assumption here: can get exact values of $F(x)$, $\partial F(x)/\partial x_e$.

Submodular Maximization via a Relaxation

Main approach:

1. Solve relaxation (approximately)

2. Round the fractional solution (approximately).

Rounding: Can be done in various ways. Many times with no/small loss (not in this talk …)

For example, in unconstrained case simply round independently.

Here: How well can we solve the multilinear relaxation?

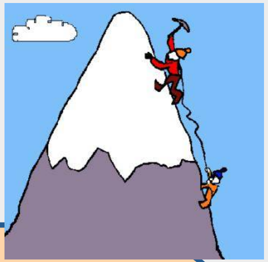
Approximating the Multilinear Relaxation

Down closed polytope: $0 \leq x \leq y,\ y \in P \rightarrow\ x \in P$.

Examples: Cardinality constraint, Matroids, Packing …

Approximating F over down closed polytopes	
Algorithms	**Hardness (value oracle)**
• 0.325 [Chekuri, Vondrak, Zenklusen'11] • $1/e \approx 0.367$ [Feldman, Naor, Schwartz'11] • 0.372 [Ene, Lê Nguyễn '16] • 0.385 [B., Feldman '17]	• 0.5 [Feige, Mirrokni, Vondrak '07] • 0.478 [Oveis Gharan, Vondrak'11]
(Monotone case): $1-1/e \approx 0.63$	
[Vondrak '08]	[Nemhauser et al. '78], [Feige '98]

Measured Continuous Greedy



Measured Continuous Greedy **[Feldman, Naor, Schwartz'11]:**

Start with an "empty solution" ($y(0)=0$)

For each $t\in[0,1]$:

$$x(t)=\arg\max_{x\in P}\{[F(y(t)\vee 1_e)-F(y(t))]\cdot x\}$$

$$dy_e(t)/dt=(1-y_e(t))\cdot x_e(t)$$

Return $y(1)$.

Remark: Linear optimization over P (P is solvable).

Analysis

Measured Continuous Greedy:

- $x(t) = \arg\max_{x \in P} \{[F(y(t) \vee 1_e) - F(y(t))] \cdot x\}$
- $dy_e(t)/dt = (1 - y_e(t)) \cdot x_e(t)$

Analysis (feasibility):

- $x(t) \in P$ ➔ Vector whose coordinate at e equals $(1 - y_e(t)) \cdot x_e(t)$ is in P (down closed).
- Output is a convex combination of points in P.

Analysis

Measured Continuous Greedy:

- $x(t) = \arg\max_{x \in P} \{[F(y(t) \vee 1_e) - F(y(t))] \cdot x\}$
- $\frac{dy_e(t)}{dt} = (1 - y_e(t)) \cdot x_e(t)$

Analysis (approximation):

$$\frac{dF(y(t))}{dt} = \sum_{e \in N} \frac{dy_e(t)}{dt} \cdot \frac{\partial F(y(t))}{\partial y_e} = \sum_{e \in N} (1 - y_e(t)) \cdot x_e(t) \frac{\partial F(y(t))}{\partial y_e}$$

$$= \sum_{e \in N} [F(y(t) \vee 1_e) - F(y(t))] \cdot x_e(t)$$

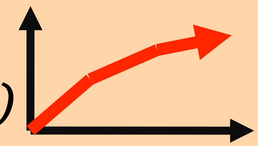
$$\geq \sum_{e \in OPT} [F(y(t) \vee 1_e) - F(y(t))] \geq F(y(t) \vee 1_{OPT}) - F(y(t))$$

Analysis (cont.)

$dF(y(t))/dt \ge F(y(t) \vee 1_{OPT}) - F(y(t))$

f monotone:

$dF(y(t))/dt \ge F(y(t) \vee 1_{OPT}) - F(y(t)) \ge f(OPT) - F(y(t))$



Solving the differential equation: $F(y(1)) \ge (1 - 1/e) f(OPT)$

f non-monotone:

Lemma 1: $F(x \vee 1_{OPT}) \ge (1 - \max_{e \in N} \{x_e\}) f(OPT)$

Lemma 2: $\forall e \in N, y_e(t) \le 1 - e^{-t}$

➔ $dF(y(t))/dt \ge F(y(t) \vee 1_{OPT}) - F(y(t)) \ge e^{-t} \cdot f(OPT) - F(y(t))$

➔ $F(y(1)) \ge 1/e \cdot f(OPT)$

Improving over 1/e

First idea: Continuous Local Search (gradient ascent)

Theorem **[Chekuri, Vondrak, Zenklusen`11]:**

Algorithm outputs vector z such that (almost):

$F(z) \geq 1/2\,[F(z \vee 1_{OPT}) + F(z \wedge 1_{OPT})]$

Assume for simplicity that z is integral.

Option 1: z is a good solution (compared to OPT).

Option 2: z is NOT a good solution.

Improving over 1/e

$F(z) \geq 1/2\,[F(z \vee 1_{OPT}) + F(z \wedge 1_{OPT})]$

- $F(z \wedge 1_{OPT})$ **is small:** z does not contain a lot of OPT's value.
- $F(z \vee 1_{OPT})$ **is small:** adding z to OPT decreases its value.

Conclusion: Avoiding the elements of z (at least somewhat) may be a good idea …

The Combined Algorithm

Combined algorithm:

- Run Continuous local search ➔ outputs z.
- Run Measured continuous greedy:

  At time $t \in [0, t_s]$: Avoid raising elements of z.

  (even if they “look good” right now)

  At time $t \in [t_s, 1]$: Continue as usual.

  Final output: x.

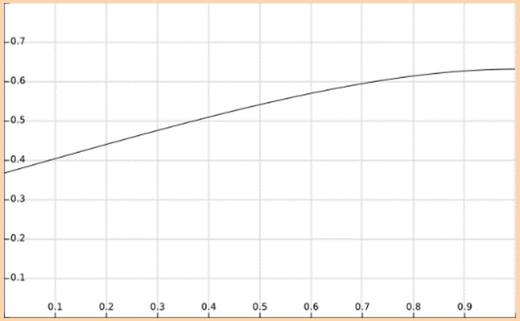
➔ Output $max\{F(z), F(x)\}$.

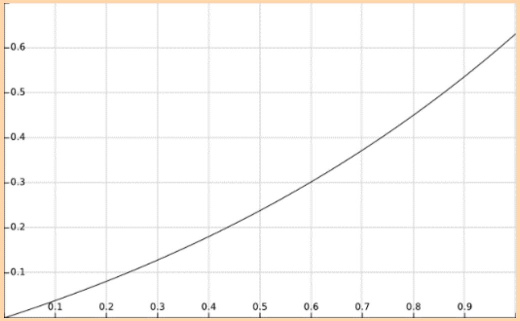
Analysis

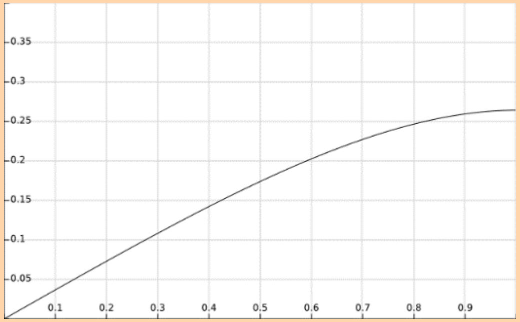
Theorem **[B., Feldman `17]:** MCG with z and parameter t_s:

$$F(x) \geq \alpha(t_s) \cdot f(OPT) - \beta(t_s) \cdot F(z \wedge 1_{OPT}) - \gamma(t_s) \cdot F(z \vee 1_{OPT})$$

- $\alpha(t_s) = e^{t_s - 1} \cdot (2 - t_s - e^{-t_s})$ $\quad | \; \alpha(0) = 1/e, \; \alpha(1) = 1 - 1/e$
- $\beta(t_s) = e^{t_s - 1} \cdot (1 - e^{-t_s})$
- $\gamma(t_s) = e^{t_s - 1} \cdot (2 - t_s - 2e^{-t_s})$







Analysis

Theorem **[B., Feldman `17]:** MCG with z and parameter t_s:

$$F(x) \geq \alpha(t_s) \cdot f(OPT) - \beta(t_s) \cdot F(z \wedge 1_{OPT}) - \gamma(t_s) \cdot F(z \vee 1_{OPT})$$

Recall: $F(z) \geq 1/2\, [F(z \vee 1_{OPT}) + F(z \wedge 1_{OPT})]$

Observation:

$$\forall 0 \leq p \leq 1,\ Max\{F(x), F(z)\} \geq p \cdot F(z) + (1-p) F(x)$$

Optimizing: with (prob.) $p = 0.23$, $t_s = 0.372$:

$$Max\{F(x), F(z)\} \geq p \cdot F(z) + (1-p) F(x) \geq 0.385 \cdot f(OPT)$$

Open Problems/Research Directions

Better approximation ratio

- For down-closed $\mathcal{P}$: close the gap $[0.385, 0.478]$.
- Cardinality constraint: $[0.385, 0.491]$.

Randomization:

Multilinear relaxation algorithms appear deterministic, but they are randomized. Evaluating the multilinear extension F requires sampling.

Can we avoid randomization?

Open Problems/Research Directions

Efficiency:

Multilinear relaxation algorithms require many function evaluations because:

- They have to make small steps to simulate continuity.
- Approximating the multilinear extension requires many samples.

Can we design faster algorithms?

Thank you

Questions?