Fast Algorithms for Maximization of Submodular Functions

Jan Vondrák[1]

[1]Theory Group
IBM Almaden Research Center

IMA Workshop — Convexity and Optimization, February 2015

Algorithms for maximization of submodular functions

What do we want to do?

Maximize a submodular function $f(S)$ over $S \in \mathcal{F}$
where $\mathcal{F}$ is a *constraint* (family of feasible sets).

Typical constraints:

- $|S| \leq k$ (influence on social networks, sensor placement, active set selection, ranking, summarization,...)
- $S \in$ matroid (welfare maximization in combinatorial auctions, above applications with group constraints)
- $\sum_{j \in S} a_{ij} \leq B_i$ (budget constraits)

Quality of solution vs. performance:

- Some simple (greedy) algorithms already work well and are fast.
- More sophisticated tools have been developed in the last decade.
- These find solutions of **better value** / for **more general constraints** but they are **slower**.

Overview of algorithms for submodular maximization

Algorithm	year	constraint	runtime	issues
Greedy	1978	cardinality	$O(kn)$	not versatile
Cont. Greedy	2008	matroid	$O(n^8)$	versatile but slow
Rnd. Local Search	2011	matroid	$O(n^5)$	still slow
DoubleGreedy	2012	none	$O(n)$	special-purpose

- Greedy works only for monotone functions and simple constraints; $O(kn)$ is fast but in some applications even this is not enough.
- Continuous Greedy gives a versatile framework, but is really slow due to random sampling and small steps in a continuous domain.

Overview of algorithms for submodular maximization

Algorithm	year	constraint	runtime	issues
Greedy	1978	cardinality	$O(kn)$	not versatile
Cont. Greedy	2008	matroid	$O(n^8)$	versatile but slow
Rnd. Local Search	2011	matroid	$O(n^5)$	still slow
DoubleGreedy	2012	none	$O(n)$	special-purpose

- Greedy works only for monotone functions and simple constraints; $O(kn)$ is fast but in some applications even this is not enough.
- Continuous Greedy gives a versatile framework, but is really slow due to random sampling and small steps in a continuous domain.

NEW:

Randomized/Thresholding Greedy	2014	cardinality	$\tilde{O}(n)$
Accelerated Cont. Greedy	2014	matroid	$\tilde{O}(n^2)$
Cont. Greedy with MWU	2015	$Ax \leq b$	$\tilde{O}(n^2)$

New algorithms in more detail

Faster algorithms: [Ashwinkumar B.V., J.V., in *SODA '14*]

- Thresholding Greedy, $O(\frac{n}{\epsilon} \log \frac{n}{\epsilon})$ time for a cardinality constraint,
- Accelerated Cont.Greedy, $O(\frac{kn}{\epsilon^4} \log^2 \frac{n}{\epsilon})$ for a matroid of rank k,

both achieving $(1 - 1/e - \epsilon)$-approximation of the optimum.

New algorithms in more detail

Faster algorithms: [Ashwinkumar B.V., J.V., in *SODA '14*]

- Thresholding Greedy, $O(\frac{n}{\epsilon} \log \frac{n}{\epsilon})$ time for a cardinality constraint,
- Accelerated Cont.Greedy, $O(\frac{kn}{\epsilon^4} \log^2 \frac{n}{\epsilon})$ for a matroid of rank k,

both achieving $(1 - 1/e - \epsilon)$-approximation of the optimum.

MWU framework: [C. Chekuri, T.S. Jayram, J.V., in *ITCS '15*]

- Continuous Greedy integrated with multiplicative weight updates
- effectively combines multiple constraints into one
- $\tilde{O}(\frac{1}{\epsilon^2}(m+n)^2)$ algorithm for m packing constraints $Ax \leq b$

New algorithms in more detail

Faster algorithms: [Ashwinkumar B.V., J.V., in *SODA '14*]

- Thresholding Greedy, $O(\frac{n}{\epsilon} \log \frac{n}{\epsilon})$ time for a cardinality constraint,
- Accelerated Cont.Greedy, $O(\frac{kn}{\epsilon^4} \log^2 \frac{n}{\epsilon})$ for a matroid of rank k,

both achieving $(1 - 1/e - \epsilon)$-approximation of the optimum.

MWU framework: [C. Chekuri, T.S. Jayram, J.V., in *ITCS '15*]

- Continuous Greedy integrated with multiplicative weight updates
- effectively combines multiple constraints into one
- $\tilde{O}(\frac{1}{\epsilon^2}(m+n)^2)$ algorithm for m packing constraints $Ax \leq b$

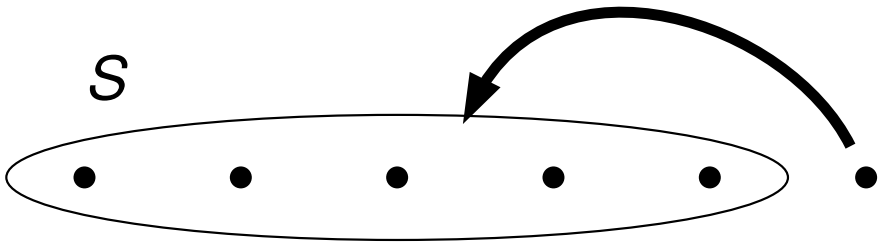
Implementation and testing of fast algorithms:
[Ashwin B.V., B. Mirzasoleiman, A. Karbasi, A. Krause, J.V., in *AAAI '15*]

- new Randomized Greedy, compared with other greedy heuristics
- datasets from *active set selection* and *k-medoid clustering*
- provable guarantees *and* demonstrated practical improvements

The Randomized Greedy Algorithm

The Greedy Algorithm: [Nemhauser-Wolsey-Fisher 1978]

- We want to solve $\max\{f(S) : |S| \leq k\}$, f monotone submodular.
- Pick elements one-by-one, maximizing the gain in $f(S)$.
- Finds a solution of value at least $(1 - 1/e)\times$ optimal.

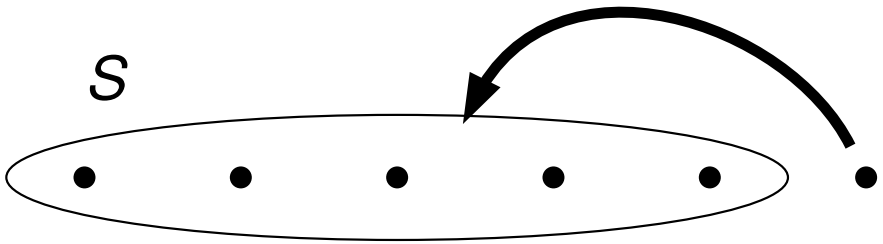


i, maximizing $f(S + i) - f(S)$

The Randomized Greedy Algorithm

The Greedy Algorithm: [Nemhauser-Wolsey-Fisher 1978]

- We want to solve $\max\{f(S) : |S| \leq k\}$, f monotone submodular.
- Pick elements one-by-one, maximizing the gain in $f(S)$.
- Finds a solution of value at least $(1 - 1/e)\times$ optimal.



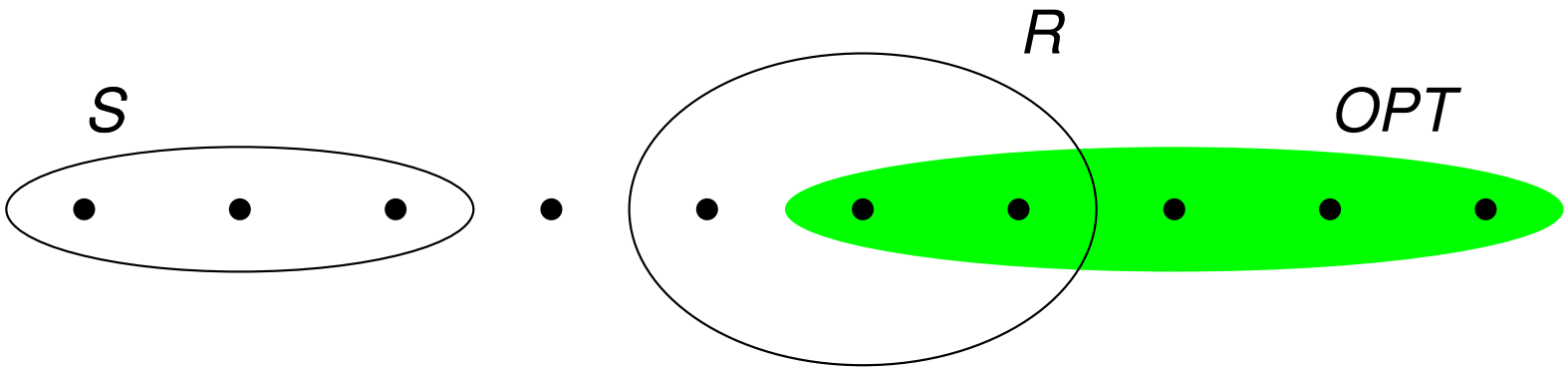
i, maximizing $f(S + i) - f(S)$

Randomized Greedy:

- Find an element i maximizing the marginal gain among all elements in a *randomly sampled set* R.
- $|R| \simeq \frac{n}{k} \log \frac{1}{\epsilon}$ is sufficient to find a solution of expected value at least $(1 - 1/e - \epsilon)\times$ optimal.
- Instead of $O(kn)$, the running time is $O(n \log \frac{1}{\epsilon})$.

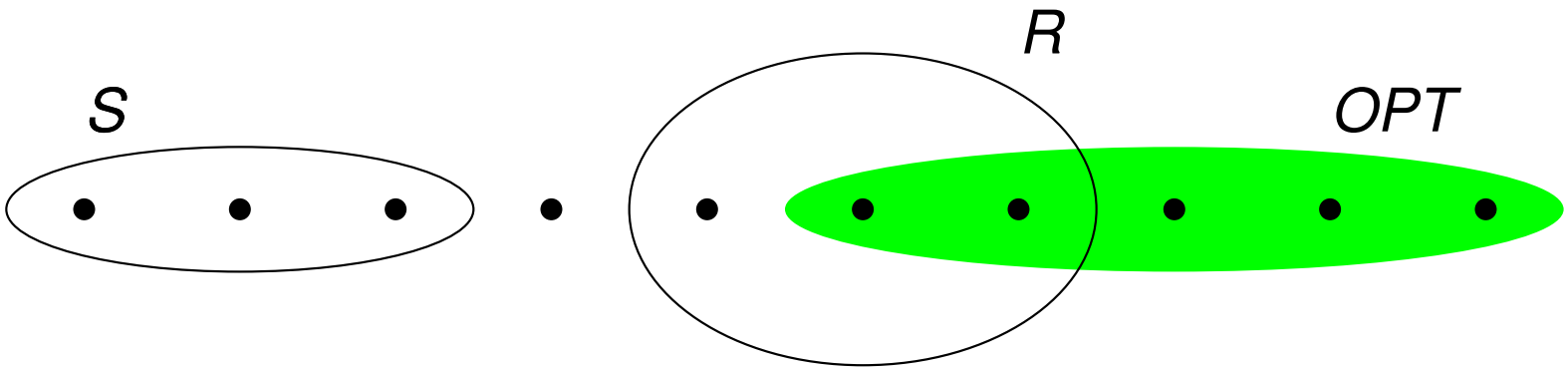
Analysis of Randomized Greedy

$S =$ current solution
$OPT =$ optimal solution (size k),
$R =$ random set (size r):



Analysis of Randomized Greedy

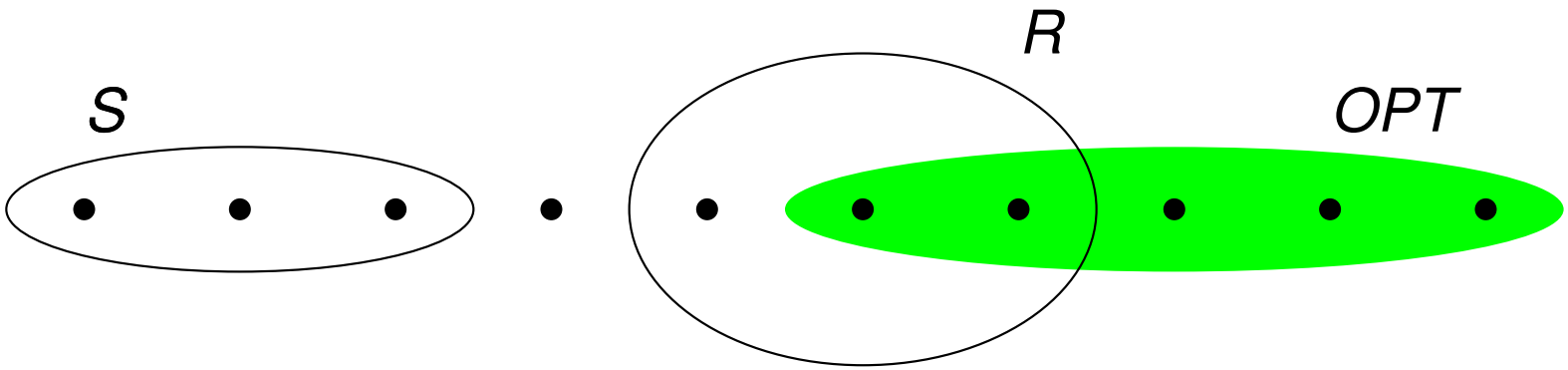
S = current solution
OPT = optimal solution (size k),
R = random set (size r):



- $\Pr[R \cap (OPT \setminus S) \neq \emptyset] \geq 1 - (1 - \frac{|OPT \setminus S|}{n})^r \geq \frac{|OPT \setminus S|}{k}(1 - e^{-rk/n})$

Analysis of Randomized Greedy

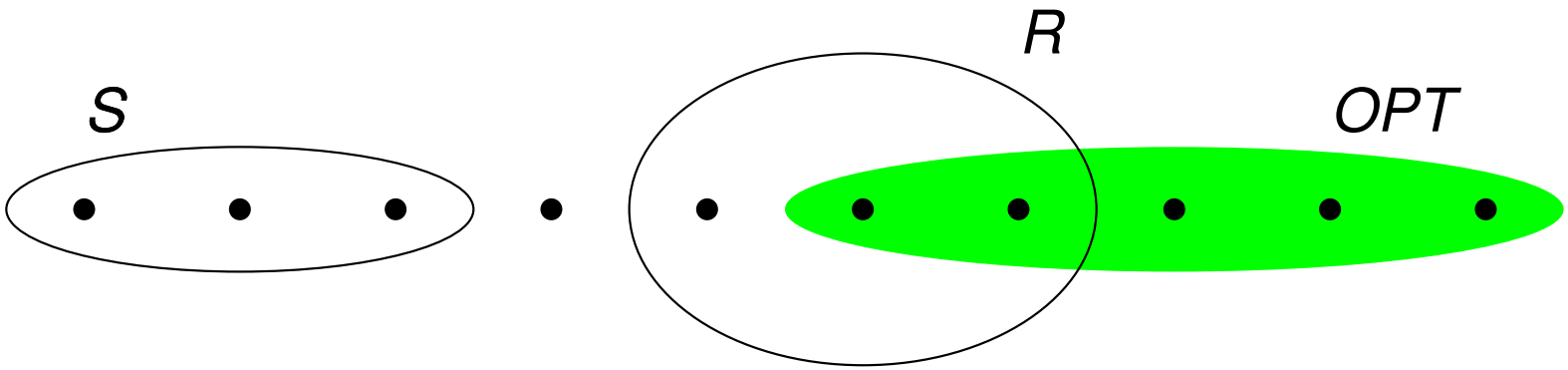
S = current solution
OPT = optimal solution (size k),
R = random set (size r):



- $\Pr[R \cap (OPT \setminus S) \neq \emptyset] \geq 1 - (1 - \frac{|OPT \setminus S|}{n})^r \geq \frac{|OPT \setminus S|}{k}(1 - e^{-rk/n})$
- We choose $r = \frac{n}{k} \log \frac{1}{\epsilon} \Rightarrow \Pr[R \cap (OPT \setminus S) \neq \emptyset] \geq (1 - \epsilon)\frac{|OPT \setminus S|}{k}$.

Analysis of Randomized Greedy

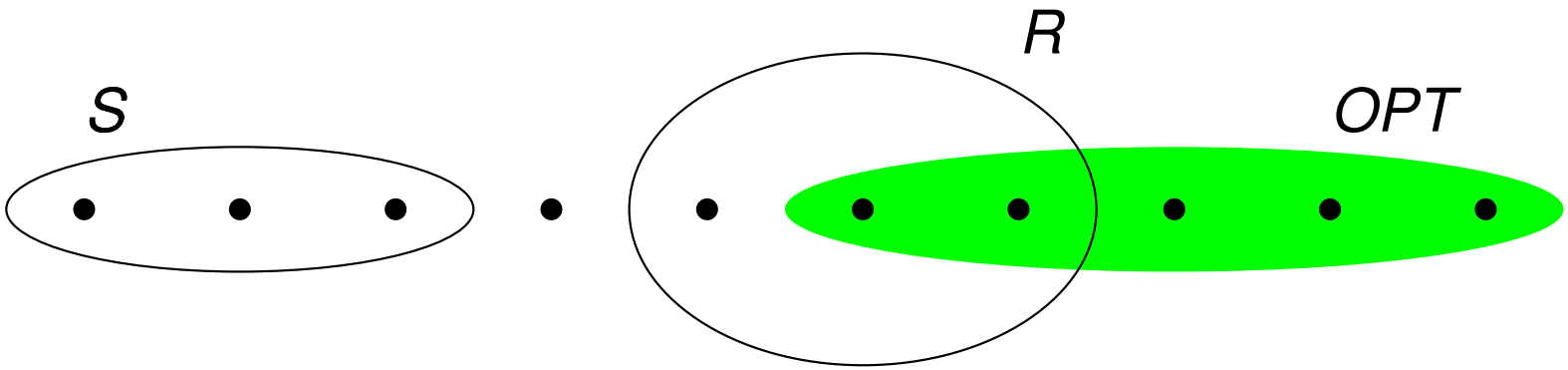
S = current solution
OPT = optimal solution (size k),
R = random set (size r):



- $\Pr[R \cap (OPT \setminus S) \neq \emptyset] \geq 1 - (1 - \frac{|OPT \setminus S|}{n})^r \geq \frac{|OPT \setminus S|}{k}(1 - e^{-rk/n})$
- We choose $r = \frac{n}{k} \log \frac{1}{\epsilon} \Rightarrow \Pr[R \cap (OPT \setminus S) \neq \emptyset] \geq (1 - \epsilon)\frac{|OPT \setminus S|}{k}$.
- Conditioned on $R \cap (OPT \setminus S) \neq \emptyset$, we gain the best element in $R \cap (OPT \setminus S)$: on average at least a random element of $OPT \setminus S$.

Analysis of Randomized Greedy

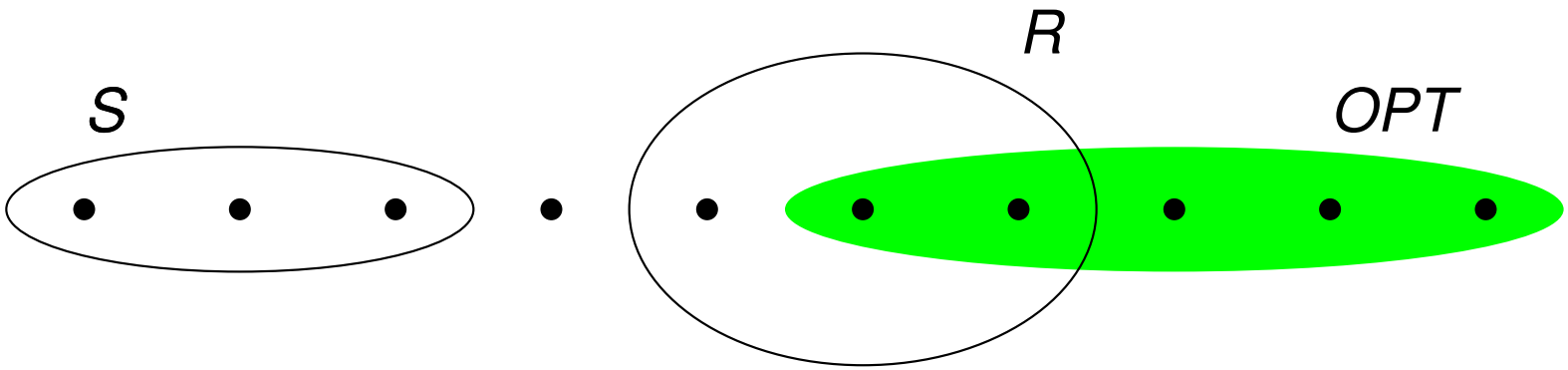
S = current solution
OPT = optimal solution (size k),
R = random set (size r):



- $\Pr[R \cap (OPT \setminus S) \neq \emptyset] \geq 1 - (1 - \frac{|OPT \setminus S|}{n})^r \geq \frac{|OPT \setminus S|}{k}(1 - e^{-rk/n})$
- We choose $r = \frac{n}{k} \log \frac{1}{\epsilon} \Rightarrow \Pr[R \cap (OPT \setminus S) \neq \emptyset] \geq (1 - \epsilon)\frac{|OPT \setminus S|}{k}$.
- Conditioned on $R \cap (OPT \setminus S) \neq \emptyset$, we gain the best element in $R \cap (OPT \setminus S)$: on average at least a random element of $OPT \setminus S$.
- $\mathbb{E}[\text{gain}] \geq \Pr[R \cap (OPT \setminus S) \neq \emptyset] \cdot \mathbb{E}_{j \in OPT \setminus S}[f_S(j)] \geq \frac{1-\epsilon}{k}(OPT - f(S))$.

Analysis of Randomized Greedy

S = current solution
OPT = optimal solution (size k),
R = random set (size r):



- $\Pr[R \cap (OPT \setminus S) \neq \emptyset] \geq 1 - (1 - \frac{|OPT \setminus S|}{n})^r \geq \frac{|OPT \setminus S|}{k}(1 - e^{-rk/n})$
- We choose $r = \frac{n}{k} \log \frac{1}{\epsilon} \Rightarrow \Pr[R \cap (OPT \setminus S) \neq \emptyset] \geq (1 - \epsilon)\frac{|OPT \setminus S|}{k}$.
- Conditioned on $R \cap (OPT \setminus S) \neq \emptyset$, we gain the best element in $R \cap (OPT \setminus S)$: on average at least a random element of $OPT \setminus S$.
- $\mathbb{E}[\text{gain}] \geq \Pr[R \cap (OPT \setminus S) \neq \emptyset] \cdot \mathbb{E}_{j \in OPT \setminus S}[f_S(j)] \geq \frac{1-\epsilon}{k}(OPT - f(S))$.

Standard greedy analysis $\Rightarrow \mathbb{E}[ALG] \geq (1 - e^{-(1-\epsilon)})OPT$.

Implementation and experiments

We implemented several algorithms for $\max\{f(S) : |S| \leq k\}$

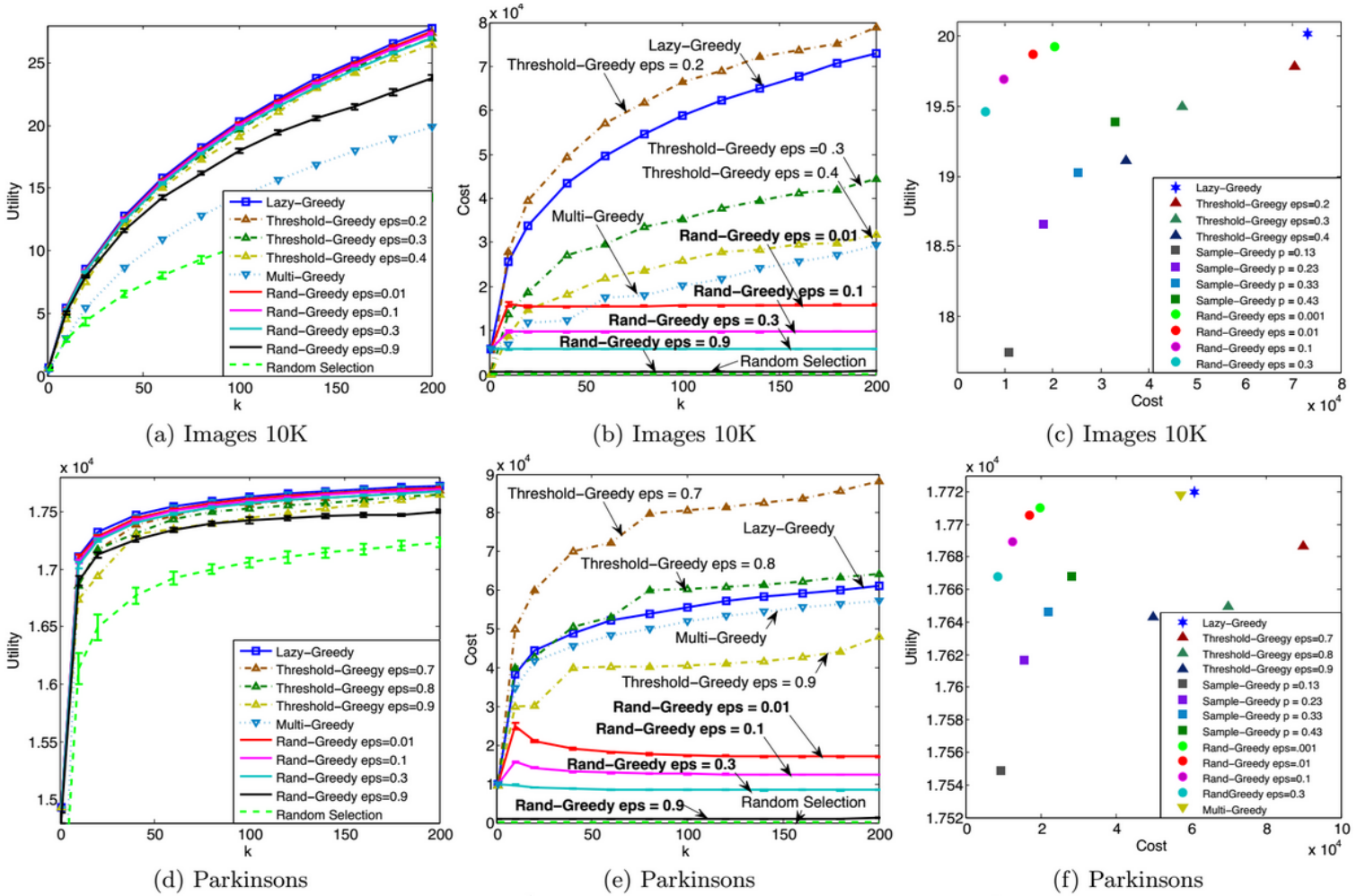
[Mirzasoleiman, Karbasi, Krause, Ashwinkumar & V.; AAAI '15]

- classical Greedy
- Lazy Greedy (speed-up heuristic, performance equiv. to Greedy)
- Sample Greedy (Greedy on a random subsample)
- Thresholding Greedy (new algorithm)
- Randomized Greedy (new algorithm)

Datasets:

1. *Active set selection:* Parkinson's Telemonitoring Dataset (5875 patients; selecting a small subset representing the dataset).
2. *k-medoid clustering:* Tiny Images, distance defined by similarity (selecting a representative subset of 10,000 images of 32x32 pixels).

Results



Key points: Rand-Greedy is the fastest method and finds solutions close to Lazy-Greedy, the top-quality but slower performer; Rand-Greedy dominates the bi-criteria plot.

Beyond the cardinality constraint

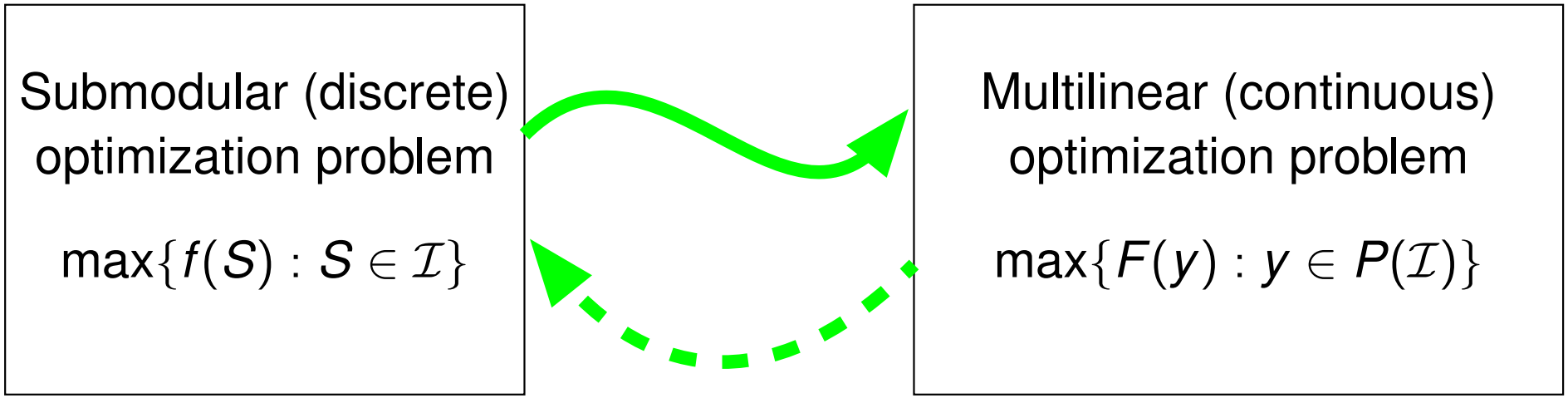
Consider $\max\{f(S) : S \in \mathcal{I}\}$ for a more complicated constraint $\mathcal{I}$.

- Greedy does not work as well anymore (depending on $\mathcal{I}$)
- We want a more robust tool — continuous relaxation!

Beyond the cardinality constraint

Consider $\max\{f(S) : S \in \mathcal{I}\}$ for a more complicated constraint $\mathcal{I}$.

- Greedy does not work as well anymore (depending on $\mathcal{I}$)
- We want a more robust tool — continuous relaxation!



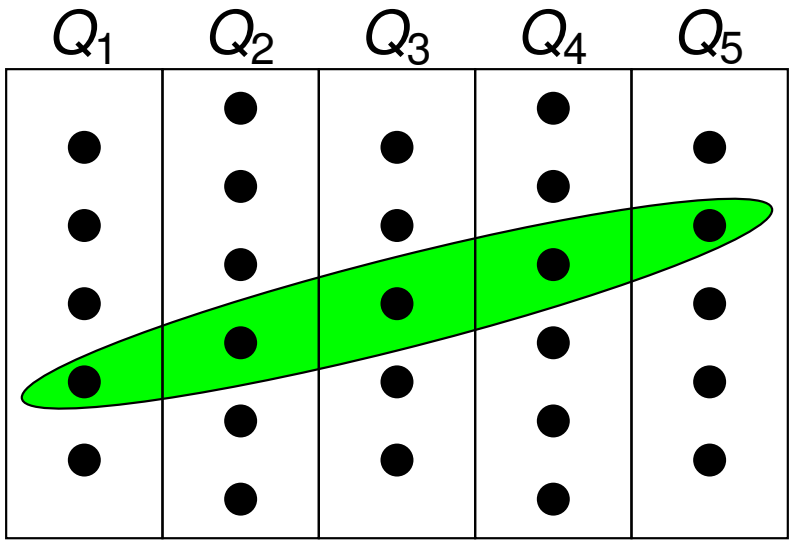
- Instead of the original problem, we solve a continuous optimization problem (approximately!)
- Multilinear relaxation is useful due to convexity/concavity properties [V. '08]
- Fractional solutions need to be rounded but that can be done for a variety of constraints; e.g. matroids [CCPV '07, CVZ '11, CVZ '12]

Matroids

Definition

A matroid on N is a system of *independent sets* $\mathcal{I} \subset 2^N$, satisfying

1. $\forall B \in \mathcal{I}, A \subset B \Rightarrow A \in \mathcal{I}$.
2. $\forall A, B \in \mathcal{I}, |A| < |B| \Rightarrow \exists x \in B \setminus A;\ A \cup \{x\} \in \mathcal{I}$.



Example: *partition matroid*

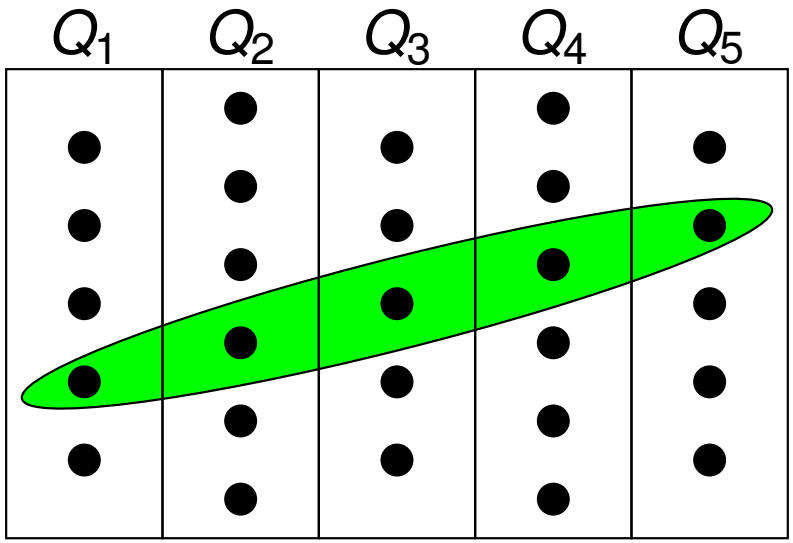
S is independent, if
$|S \cap Q_i| \leq k_i$ for each Q_i.

Matroids

Definition

A matroid on N is a system of *independent sets* $\mathcal{I} \subset 2^N$, satisfying

1. $\forall B \in \mathcal{I}, A \subset B \Rightarrow A \in \mathcal{I}$.
2. $\forall A, B \in \mathcal{I}, |A| < |B| \Rightarrow \exists x \in B \setminus A;\ A \cup \{x\} \in \mathcal{I}$.



Example: *partition matroid*

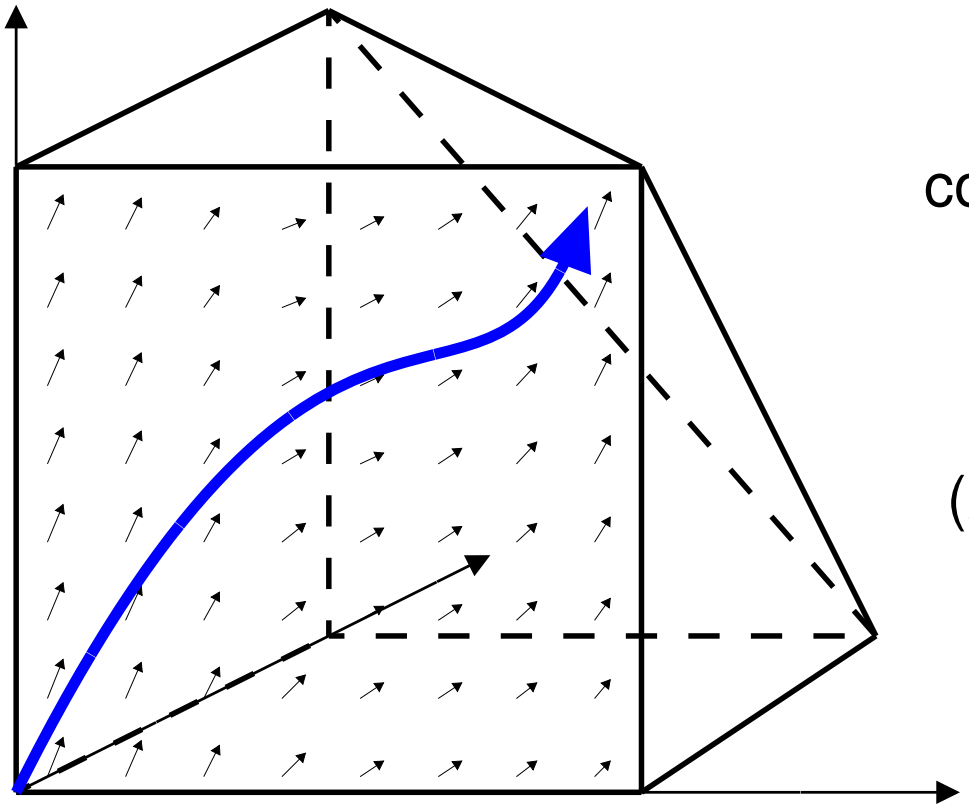
S is independent, if
$|S \cap Q_i| \leq k_i$ for each Q_i.

Multilinear relaxation works well with matroids: [Calinescu,Chekuri,Pal,V. '07]

- Any fractional solution of the problem $\max\{F(x) : x \in P(\mathcal{I})\}$ can be converted into a discrete solution $I \in \mathcal{I}$ without loss of value.

Continuous Greedy [V. '08]

Problem: continuous optimization over a polytope, $\max\{F(y) : y \in P\}$.



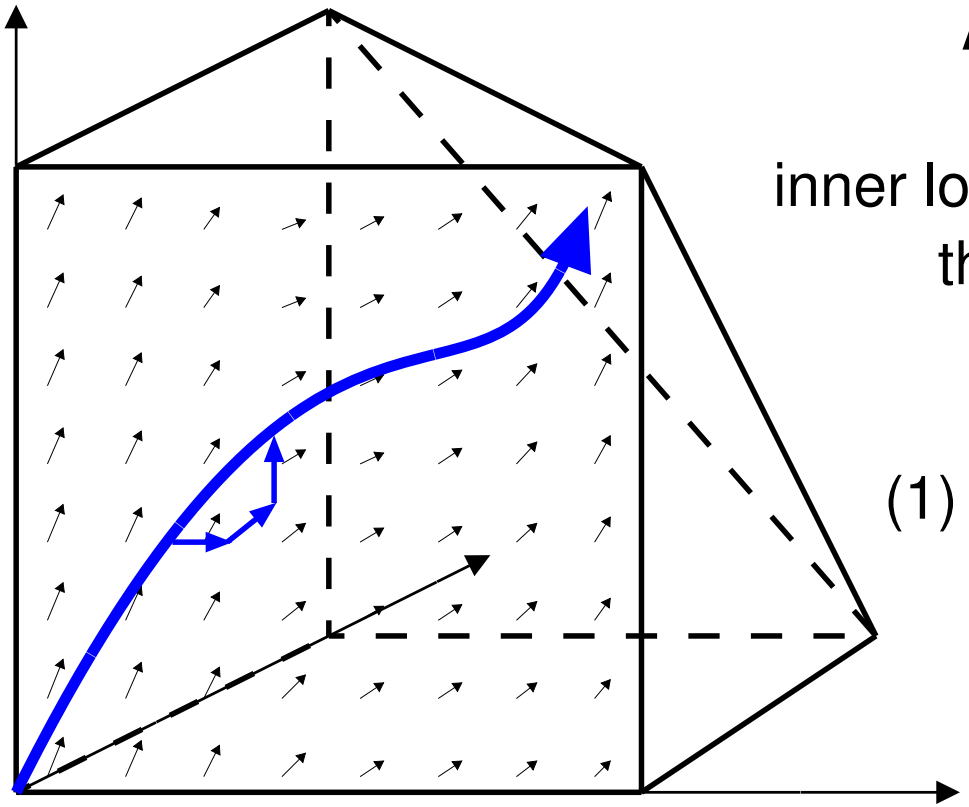
Continuous Greedy Algorithm:
$\frac{dx}{dt} = \text{argmax}_{v \in P} v \cdot \nabla F(x)$,
continuous process gets discretized.

Slow ($O(n^8)$) because
(1) small time steps are necessary
(2) $F(x)$ requires random sampling.

Approximation guarantee: $F(x(1)) \geq (1 - 1/e)OPT$.

Accelerated Continuous Greedy [Ashwinkumar-V. '14]

Problem: optimization over a matroid polytope, $\max\{F(y) : y \in P(\mathcal{I})\}$.



Accelerated Continuous Greedy:
$\frac{dx}{dt} = \text{argmax}_{v \in P} v \cdot \nabla F(x)$;
inner loop is replaced by *Discrete Greedy*,
this allows larger steps without loss.

Faster ($\tilde{O}(kn)$) because
(1) larger steps mean fewer iterations,
(2) fewer samples per iteration.

Approximation guarantee: $F(x(1)) \geq (1 - 1/e - \epsilon)OPT$.

Analysis of Accelerated Continuous Greedy (1)

Previously: find $\mathbf{v} \in P(\mathcal{I})$ maximizing $\mathbf{v} \cdot \nabla F$, update $\mathbf{x}' = \mathbf{x} + \delta\mathbf{v}$.

$$F(\mathbf{x}') - F(\mathbf{x}) \geq \delta(\mathbf{v} \cdot \nabla F) - O(\delta^2 \mathbf{v}^T (\nabla^2 F)\mathbf{v})$$
$$\geq \delta(OPT - F(\mathbf{x})) - O(\delta^2 \mathbf{v}^T (\nabla^2 F)\mathbf{v});$$

we choose $\delta = O(1/n^2)$ to make the error $\ll$ the gain.

Analysis of Accelerated Continuous Greedy (1)

Previously: find $\mathbf{v} \in P(\mathcal{I})$ maximizing $\mathbf{v} \cdot \nabla F$, update $\mathbf{x}' = \mathbf{x} + \delta\mathbf{v}$.

$$F(\mathbf{x}') - F(\mathbf{x}) \geq \delta(\mathbf{v} \cdot \nabla F) - O(\delta^2 \mathbf{v}^T (\nabla^2 F)\mathbf{v})$$
$$\geq \delta(OPT - F(\mathbf{x})) - O(\delta^2 \mathbf{v}^T (\nabla^2 F)\mathbf{v});$$

we choose $\delta = O(1/n^2)$ to make the error $\ll$ the gain.

New approach: find $\mathbf{v} = \mathbf{1}_I \in P(\mathcal{I})$ one coordinate at a time, while updating partial derivatives after each increment:

$$F(\mathbf{x}') = F(\mathbf{x}) + \delta \sum_{i=1}^{k} \frac{\partial F}{\partial x_{\pi(i)}}\Big|_{\mathbf{x}+\sum_{j<i} \delta \mathbf{e}_{\pi(j)}}.$$

Analysis of Accelerated Continuous Greedy (1)

Previously: find $\mathbf{v} \in P(\mathcal{I})$ maximizing $\mathbf{v} \cdot \nabla F$, update $\mathbf{x}' = \mathbf{x} + \delta\mathbf{v}$.

$$F(\mathbf{x}') - F(\mathbf{x}) \geq \delta(\mathbf{v} \cdot \nabla F) - O(\delta^2 \mathbf{v}^T (\nabla^2 F)\mathbf{v})$$
$$\geq \delta(OPT - F(\mathbf{x})) - O(\delta^2 \mathbf{v}^T (\nabla^2 F)\mathbf{v});$$

we choose $\delta = O(1/n^2)$ to make the error $\ll$ the gain.

New approach: find $\mathbf{v} = \mathbf{1}_I \in P(\mathcal{I})$ one coordinate at a time, while updating partial derivatives after each increment:

$$F(\mathbf{x}') = F(\mathbf{x}) + \delta \sum_{i=1}^{k} \frac{\partial F}{\partial x_{\pi(i)}}\Big|_{\mathbf{x}+\sum_{j<i} \delta \mathbf{e}_{\pi(j)}}.$$

Why is it better? Updating partial derivatives between increments leads to a cleaner analysis, mimicking the discrete greedy algorithm:

$$F(\mathbf{x}') - F(\mathbf{x}) \geq \delta \sum_{j \in OPT} \frac{\partial F}{\partial x_j}\Big|_{\mathbf{x}'} \geq \delta(OPT - F(\mathbf{x}')).$$

Analysis of Accelerated Continuous Greedy (2)

We have:

$$F(\mathbf{x}') - F(\mathbf{x}) \geq \delta \sum_{j \in OPT} \frac{\partial F}{\partial x_j}\Big|_{\mathbf{x}'} \geq \delta(OPT - F(\mathbf{x}')).$$

After $1/\delta$ steps:

$$F(\mathbf{x}_{out}) \geq \left(1 - \frac{1}{(1+\delta)^{1/\delta}}\right) OPT.$$

Analysis of Accelerated Continuous Greedy (2)

We have:

$$F(\mathbf{x}') - F(\mathbf{x}) \geq \delta \sum_{j \in OPT} \frac{\partial F}{\partial x_j}\Big|_{\mathbf{x}'} \geq \delta(OPT - F(\mathbf{x}')).$$

After $1/\delta$ steps:

$$F(\mathbf{x}_{out}) \geq \left(1 - \frac{1}{(1+\delta)^{1/\delta}}\right) OPT.$$

Notes:

- we can choose $\delta > 0$ constant, to achieve $(1 - 1/e - O(\delta))OPT$;
- even $\delta = 1$ works, to achieve $\frac{1}{2}OPT$;
- we interpolate between classical greedy and continuous greedy.

Analysis of Accelerated Continuous Greedy (2)

We have:

$$F(\mathbf{x}') - F(\mathbf{x}) \geq \delta \sum_{j \in OPT} \frac{\partial F}{\partial x_j}\Big|_{\mathbf{x}'} \geq \delta(OPT - F(\mathbf{x}')).$$

After $1/\delta$ steps:

$$F(\mathbf{x}_{out}) \geq \left(1 - \frac{1}{(1+\delta)^{1/\delta}}\right) OPT.$$

Notes:

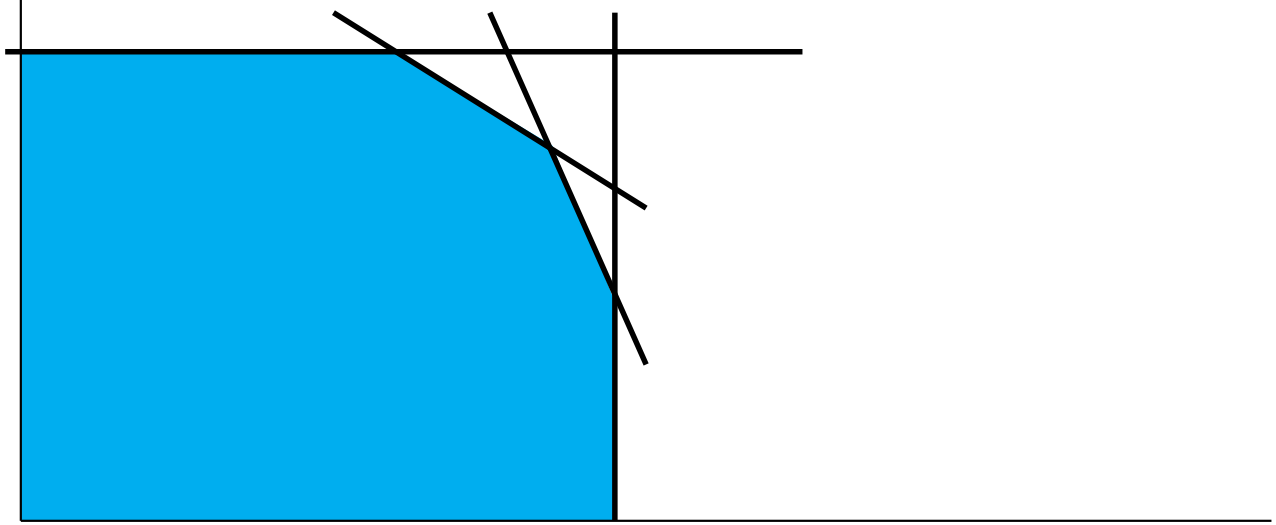
- we can choose $\delta > 0$ constant, to achieve $(1 - 1/e - O(\delta))OPT$;
- even $\delta = 1$ works, to achieve $\frac{1}{2}OPT$;
- we interpolate between classical greedy and continuous greedy.

Additional speed-up tricks lead to $O(\frac{kn}{\delta^4} \log^2 \frac{n}{\delta})$ running time for the problem $\max\{f(S) : S \in \mathcal{I}\}$ (including rounding).

Beyond matroids

Less structured constraints: suppose m constraints, n variables,

$$\max\{f(\mathbf{x}) : A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \{0,1\}^n\}.$$



- Discrete greedy doesn't work at all.
- Fractional problem $\max\{F(\mathbf{x}) : A\mathbf{x} \leq \mathbf{b}\}$ can still be useful.
- Rounding depends on the properties of A, e.g. k-sparse matrices can be handled with a loss of factor $O(k)$.

Submodular optimization with MWU

Multiplicative Weight Updates: iterative technique useful in optimization, game theory, machine learning, differential privacy, . . .

Submodular optimization with MWU:

- Discrete Greedy has been combined with MWU [Azar-Gamzu '12], performance depends on the *width* of the constraints.
- We show that Continuous Greedy can be integrated with MWU in a natural way, to obtain a $(1 - 1/e - \epsilon)$-approximation for $\max\{F(\mathbf{x}) : A\mathbf{x} \leq \mathbf{1}\}$ [Chekuri, Jayram, V.; ITCS '15]
- The framework is more versatile: allowing combinations of convex constraints + concave and (nonmonotone) submodular objectives.

Continuous-time MWU process (monotone case)

Initialize: $\mathbf{x}(0) = (0, 0, \ldots, 0), \mathbf{w}(0) = (1, 1, \ldots, 1)$.
Continuous process running from $t = 0$ to $t = 1$:

$$\begin{aligned}
\mathbf{v}(t) &= \operatorname{argmax}\{\mathbf{y} \cdot \nabla F|_{\mathbf{x}(t)} : \mathbf{y} \geq 0, \mathbf{w}^T A\mathbf{y} \leq \mathbf{w}^T \mathbf{1}\} \\
\frac{dw_i}{dt} &= \eta w_i(t) A_i \mathbf{v}(t) \qquad \forall i \in [m] \\
\frac{d\mathbf{x}}{dt} &= \mathbf{v}(t)
\end{aligned}$$

Continuous-time MWU process (monotone case)

Initialize: $\mathbf{x}(0) = (0, 0, \ldots, 0), \mathbf{w}(0) = (1, 1, \ldots, 1)$.
Continuous process running from $t = 0$ to $t = 1$:

$$\begin{aligned} \mathbf{v}(t) &= \text{argmax}\{\mathbf{y} \cdot \nabla F|_{\mathbf{x}(t)} : \mathbf{y} \geq 0, \mathbf{w}^T A\mathbf{y} \leq \mathbf{w}^T \mathbf{1}\} \\ \frac{dw_i}{dt} &= \eta w_i(t) A_i \mathbf{v}(t) \qquad \forall i \in [m] \\ \frac{d\mathbf{x}}{dt} &= \mathbf{v}(t) \end{aligned}$$

Analysis:

- $A_i\mathbf{x}(1) = \frac{1}{\eta}\int_0^1 \frac{1}{w_i(t)}\frac{dw_i}{dt}dt = \frac{1}{\eta}\int_0^1 \frac{d}{dt}(\ln w_i(t))dt = \frac{1}{\eta}\ln w_i(1)$
- $\frac{1}{\eta}\ln\sum_i w_i(1) - \frac{1}{\eta}\ln m = \frac{1}{\eta}\int_0^1 \frac{d}{dt}(\ln\sum_i w_i(t))dt = \int_0^1 \frac{\mathbf{w}^T A\mathbf{v}(t)}{\mathbf{w}^T\mathbf{1}}dt \leq 1$
- $\frac{dF}{dt} = \mathbf{v}(t) \cdot \nabla F \geq \mathbf{x}^* \cdot \nabla F \geq OPT - F(\mathbf{x}(t))$

$\Rightarrow A\mathbf{x}(1) \leq 1 + \frac{\ln m}{\eta}, \quad F(\mathbf{x}(1)) \geq (1 - 1/e)OPT.$

Discretization of the MWU process

Differential equations $\Rightarrow$ *algorithm*:

- Direction $\mathbf{v}(t)$ can be chosen to be a *single nonzero coordinate*.
- Non-uniform time steps are chosen to guarantee strongly polynomial running time (à la [Garg-Könemann '98])
- Time steps can be defined by requiring each weight to multiply by at most e^{ϵ} at a time.

Discretization of the MWU process

Differential equations $\Rightarrow$ *algorithm*:

- Direction $\mathbf{v}(t)$ can be chosen to be a *single nonzero coordinate*.
- Non-uniform time steps are chosen to guarantee strongly polynomial running time (à la [Garg-Könemann '98])
- Time steps can be defined by requiring each weight to multiply by at most e^{ϵ} at a time.
- Objective function does not pose issues since we move along coordinate-aligned steps $\Rightarrow$ $F(x)$ is linear in each step!
- More complications for non-monotone submodular functions.

Discretization of the MWU process

Differential equations $\Rightarrow$ *algorithm*:

- Direction $\mathbf{v}(t)$ can be chosen to be a *single nonzero coordinate*.
- Non-uniform time steps are chosen to guarantee strongly polynomial running time (à la [Garg-Könemann '98])
- Time steps can be defined by requiring each weight to multiply by at most e^{ϵ} at a time.
- Objective function does not pose issues since we move along coordinate-aligned steps $\Rightarrow$ $F(x)$ is linear in each step!
- More complications for non-monotone submodular functions.

Results:

1. $(1 - 1/e - \epsilon)$-approximation in time $\tilde{O}(\frac{1}{\epsilon^4}(m+n)^2)$ for $\max\{F(\mathbf{x}) : A\mathbf{x} \leq \mathbf{b}\}$ when F is monotone multilinear submodular,
2. $(1 - 1/e - \epsilon)$-approximation in time $\tilde{O}(\frac{1}{\epsilon^4}mn^2(m+n))$ for $\max\{F(\mathbf{x}) : A\mathbf{x} \leq \mathbf{b}\}$ when F is nonmonotone smooth submodular.

Conclusions

1. Submodular function maximization under the constraint $|S| \leq k$ can be done very fast (near-linear time)
2. More complicated constraints can be handled using variants of Continuous Greedy
3. Continuous Greedy is not as slow as originally designed: can be implemented in $\tilde{O}(kn)$ running time for matroids.

   (Recent improvement [Buchbinder-Feldman-Schwartz '15]: $\tilde{O}(n + k\sqrt{n})$ running time for partition matroids.)

Questions:

1. Is near-linear time possible for $(1 - 1/e)$-approx for submodular maximization over matroids?
2. Is it possible to eliminate randomization?