Massachusetts
Institute of
Technology

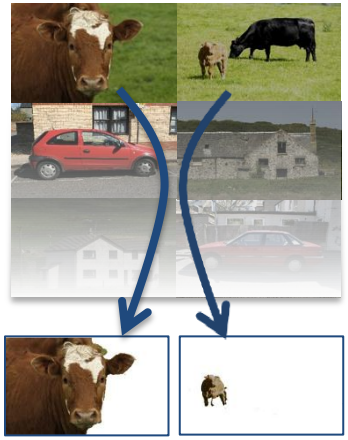
Optimizing submodular functions

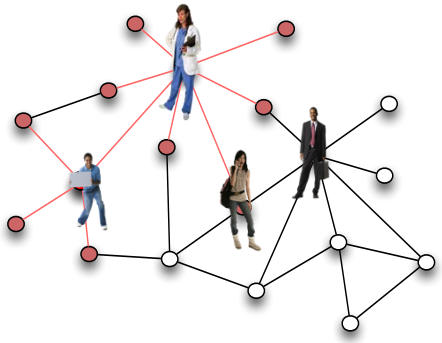
CVPR 2015 Tutorial

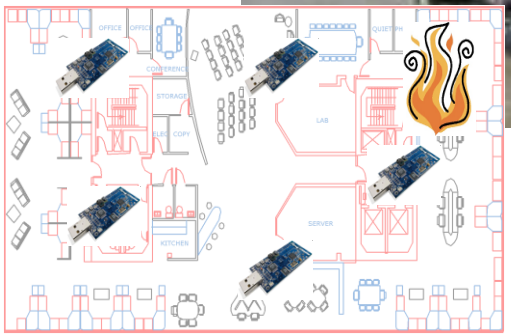
Stefanie Jegelka

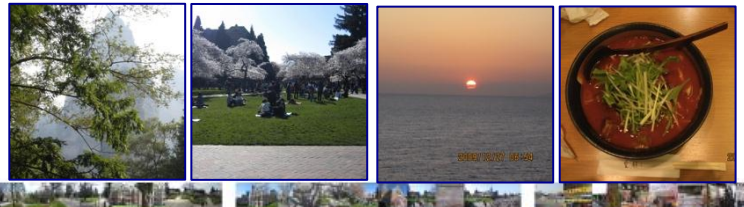
MIT

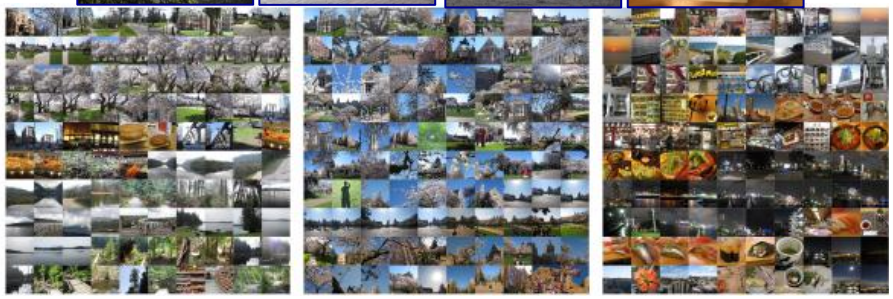
Subset selection problems



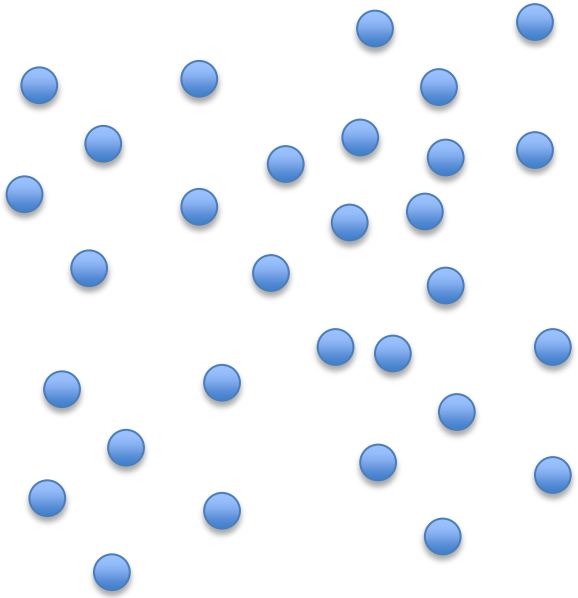






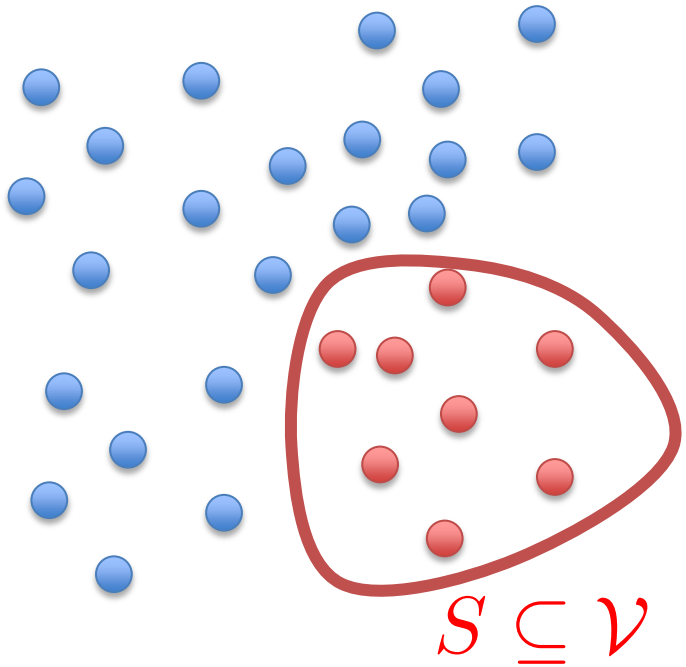


Setup



- ground set $\mathcal{V}$

Setup



- ground set $\mathcal{V}$
- (scoring) function

$$F : 2^{\mathcal{V}} \rightarrow \mathbb{R}_+$$

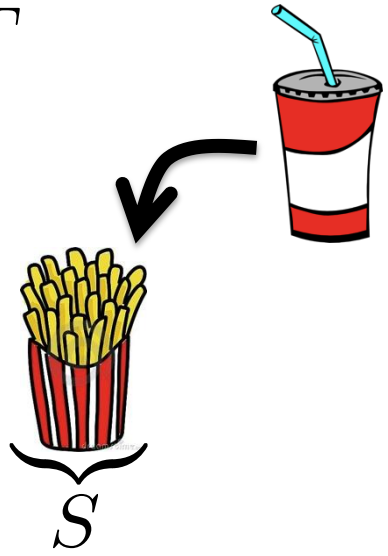
We will assume:
- $F(\emptyset) = 0$
- black box "oracle" to evaluate *F*

$$\max \quad F(S)$$

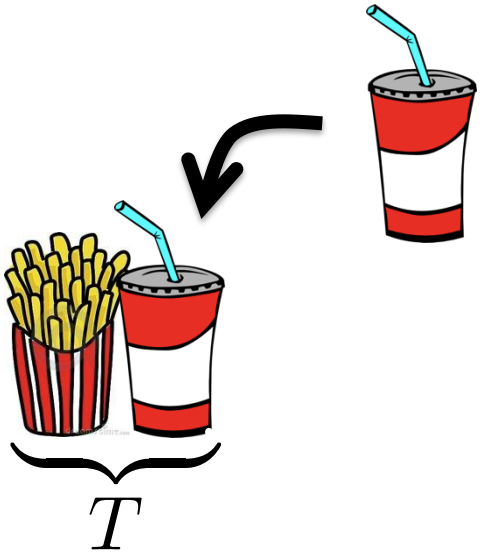
$$\min_{S \subseteq \mathcal{V}} F(S)$$

Submodularity

$S \subseteq T$



S



T

$$F(S \cup s) - F(S) \quad \geq \quad F(T \cup s) - F(T)$$

extra cost:
one drink

extra cost:
free refill ☺

diminishing marginal costs

Outline

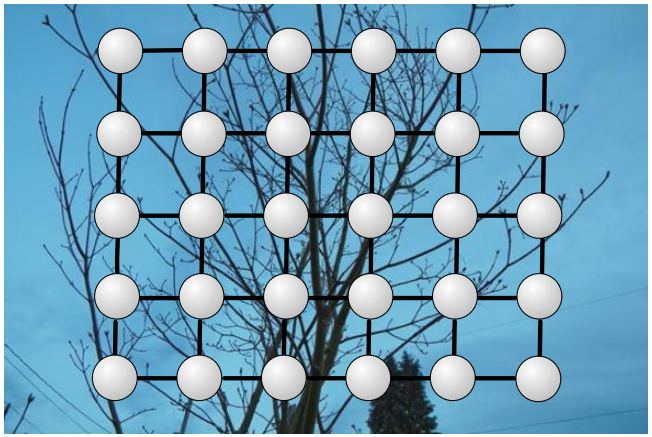
- submodularity and convexity
  - Lovasz extension: exact relaxations
  - fast algorithms
  - structured norms …

morning session

- constrained minimization

- submodular maximization
  - efficiently finding diverse, informative subsets

Recall: MAP and cuts

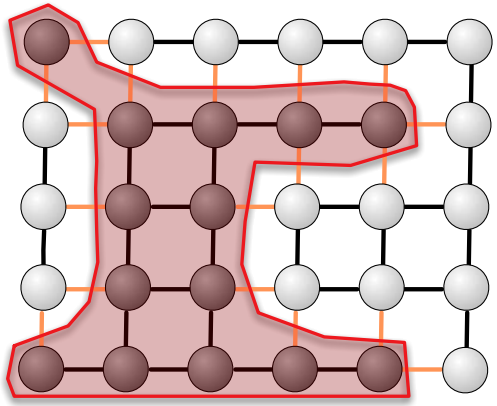


binary labeling: $x = 1_A$

pairwise random field:

$$E(x) = \mathrm{Cut}(A)$$

What's the problem?

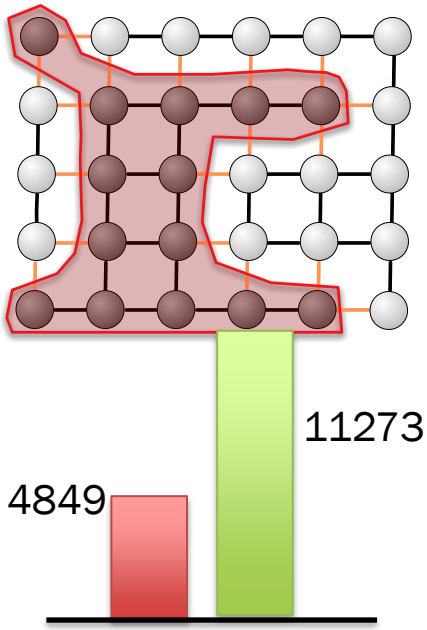


minimum cut: prefer short cut = short object boundary

What's wrong?

we get ...

local coherence
= short cut

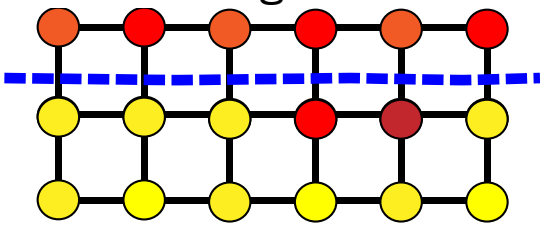


cut weight
= energy

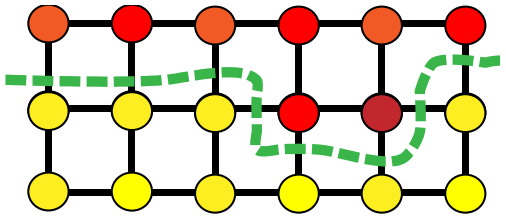
ideally ...

homogeneous cut
global dependencies!

not homogeneous



homogeneous



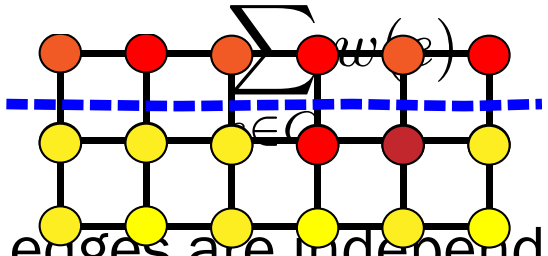
Cooperative cuts

ideally ...

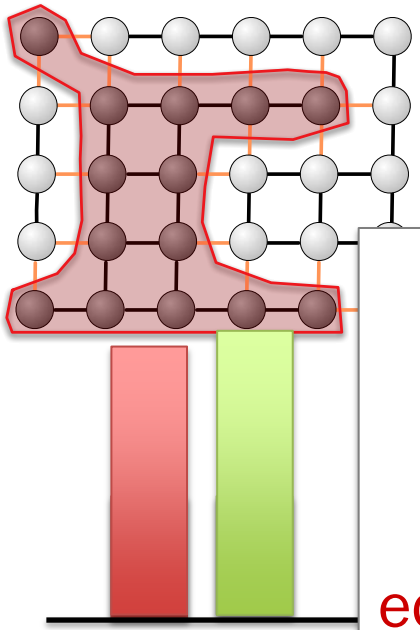
local coherence
= short cut

cost of a cut $C \subseteq \mathcal{E}$:

$$\sum_{e \in C} w(e)$$



edges are independent



cut weight
= energy

homogeneous cut
global dependencies!

cooperative graph cut

cost of a cut $C \subseteq \mathcal{E}$:

submodular function

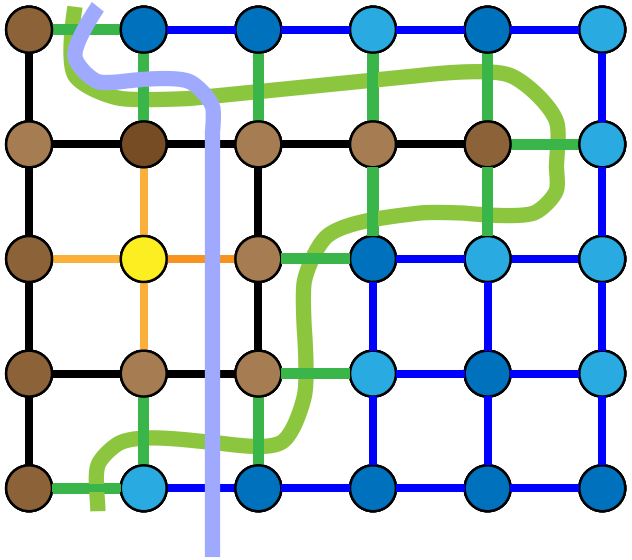
$F(C)$

edges are not independent

Homogeneity via group sparsity

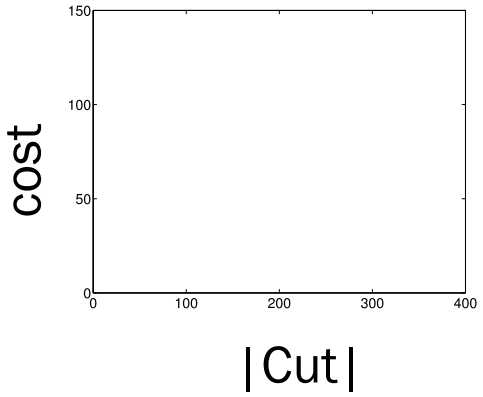
sum of weights:
use few edges

submodular cost function:
use few types of edges



One type (13 edges)
Many types (6 edges)

$$F(\text{Cut}) = \sum_{\text{type } k} F_k(\text{Cut})$$



(Jegelka & Bilmes 2011)

Results

Random Walker
[Grady 06]

Curvature Reg.
[El-Zehiry & Grady 10]

Graph Cut
[Boykov & Jolly 01]

Cooperative Cut

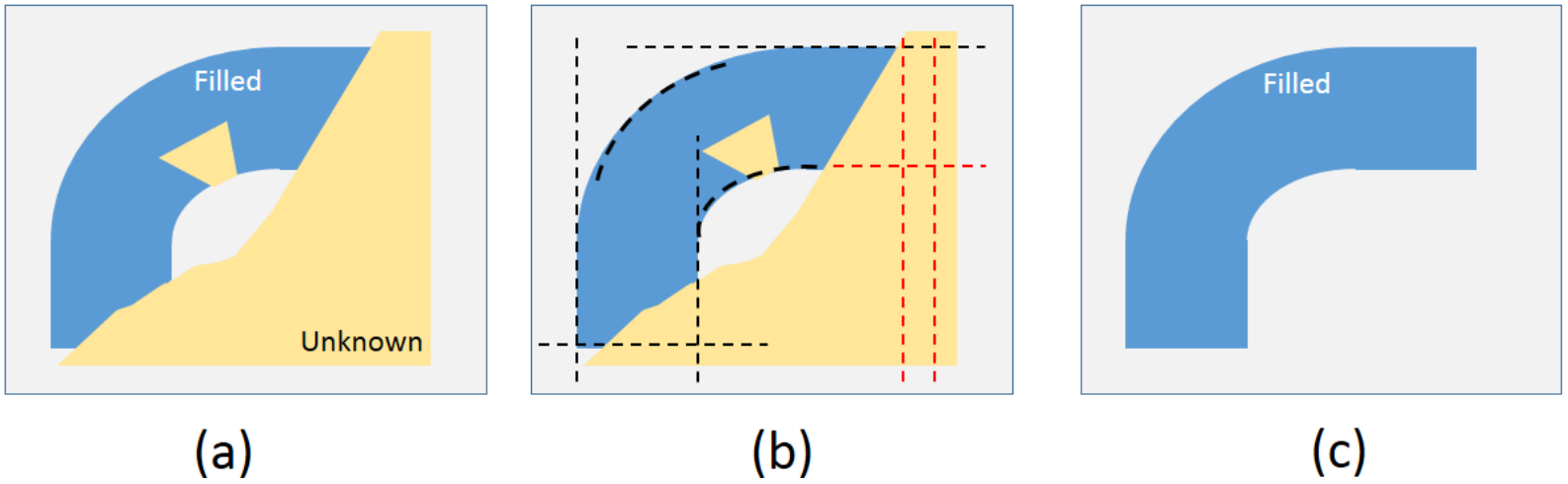
Quantitatively: up to 70% reduction in error!

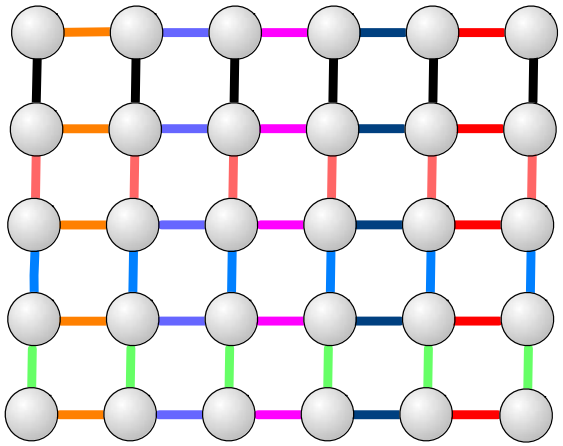
Results

Graph cut

Cooperative cut

Similarly: contour completion RF





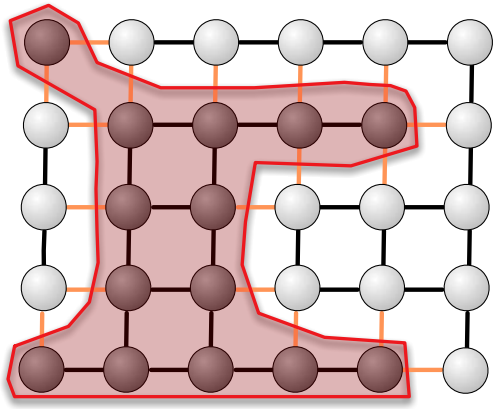
geometric edge groups:

- straight lines
- parabolas

(Silberman et al 2014)

Inference?

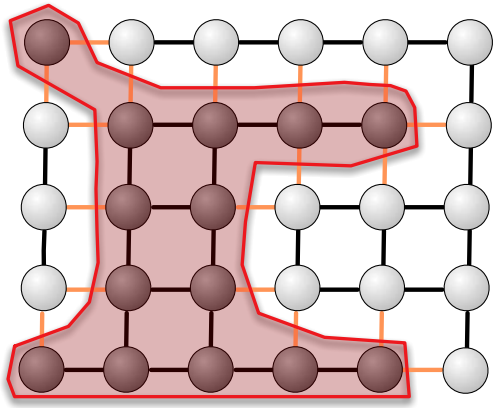
- not a submodular energy function



$$\sum_k F_k(\mathrm{Cut}(x)) = \sum_k F_k\Big(\sum_{(i,j)\in G_k} \psi_{ij}(x_i, x_j)\Big)$$

Inference?

- not a submodular energy function
- find a minimum cut $C \subseteq \mathcal{E}$
  cost function:



normally:

$$\text{Cost}(C) = \sum_{e \in C} w(e)$$

now:

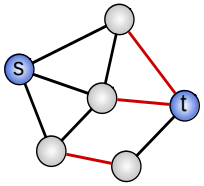
$$\text{Cost}(C) = F(C)$$

$$\min F(S) \text{ s.t. constraints on } S$$

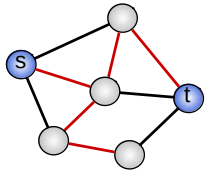
Constrained minimization

$$\min F(S) \text{ s.t. constraints on } S$$

e.g.



S is a cut



S is a spanning tree

$$|S| = k$$

usually very hard. ➔ approximations

convex relaxation (not exact!)

majorize-minimize

(Goel et al. 09, Iwata & Nagano 09, Goemans et al. 09, Jegelka & Bilmes 11, Iyer et al. ICML 13, Kohli et al 13…)

Submodularity and concavity

- submodularity:

$A \subseteq B, \;\; s \notin B$:

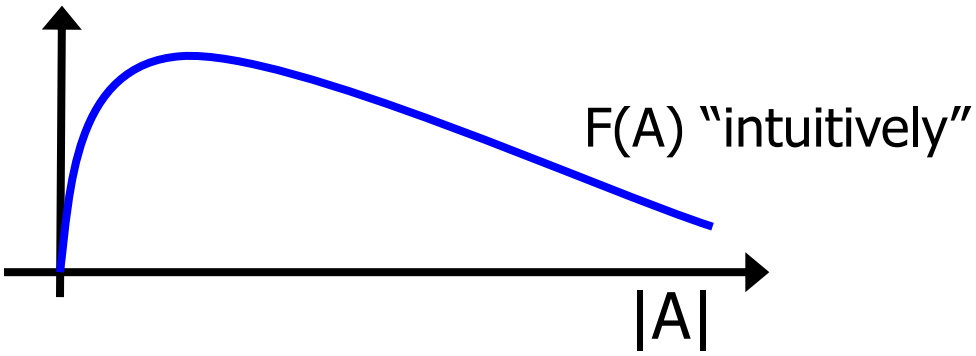
$$F(A \cup s) - F(A) \;\; \geq \;\; F(B \cup s) - F(B)$$

A +• s B +• s

- concavity:

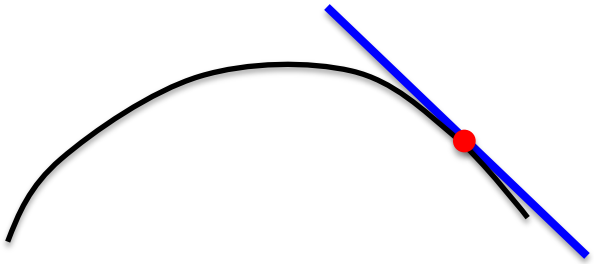
$a \leq b, \;\; s > 0$:

$$f(a + s) - f(a) \;\; \geq \;\; f(b + s) - f(b)$$



A practical algorithm

idea: submodularity = discrete concavity



"super-gradient"

For *i* = *1,2,…*

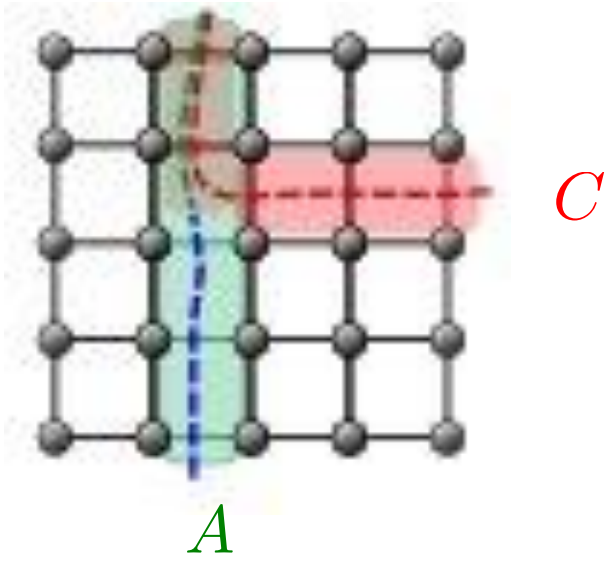
- compute linear upper bound $\widehat{F}_i$ with $\widehat{F}_i(C_i) = F(C_i)$
- find tree/path/… C_{i+1} with minimum $\widehat{F}_i(C)$.

familiar ☺
e.g. min-cut

fast: only need to solve linear optimization problem!

(Jegelka & Bilmes 2011; Iyer, Jegelka, Bilmes 2013)

Supergradient



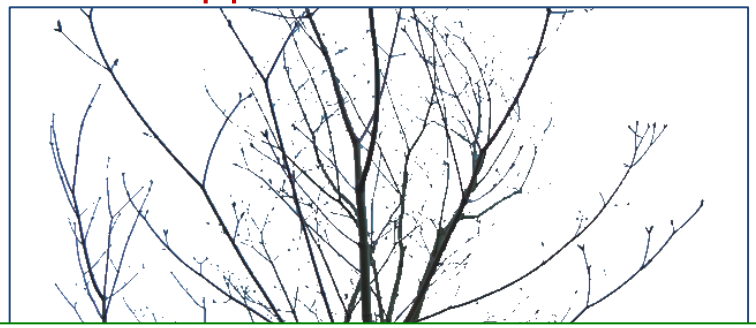
$$\widehat{F}_i(C) = F(A) + \sum_{e \in C \setminus A} F(e|A) - \sum_{e \in A \setminus C} F(e|\mathcal{E} \setminus e) \geq F(C)$$

$$F(e|A) = F(A \cup e) - F(A)$$

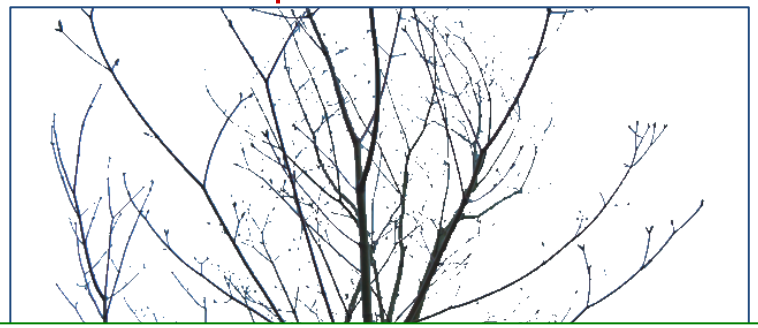
Does it work?

minimum cut solution

approximate solution



optimal solution



- usually works well in practice
- theory: approximation guarantees depending on curvature of *F*

(Iyer et al 2013)

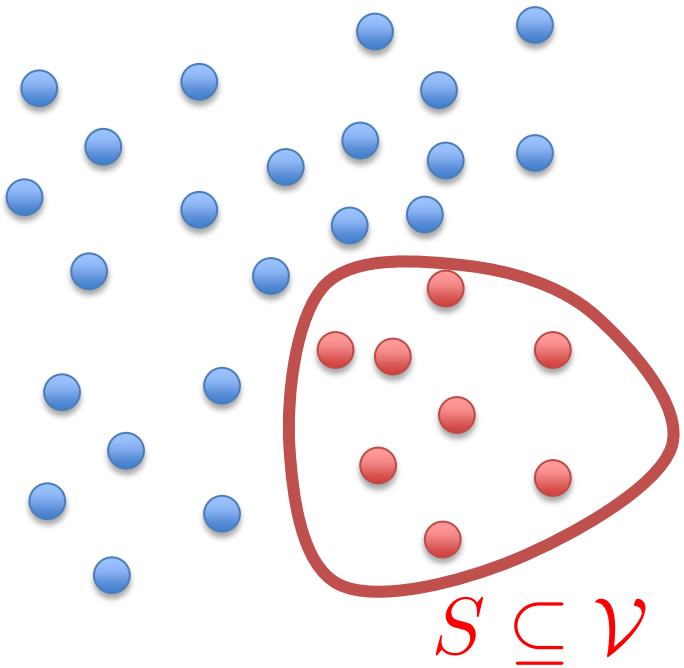
- special cases: exact solution *(Kohli et al 2013)*
- also possible for maximization!

Outline

- submodularity and convexity
  - Lovasz extension: exact relaxations
  - fast algorithms
  - structured norms …
- constrained minimization
- submodular maximization
  - efficiently finding diverse, informative subsets

morning session

Setup

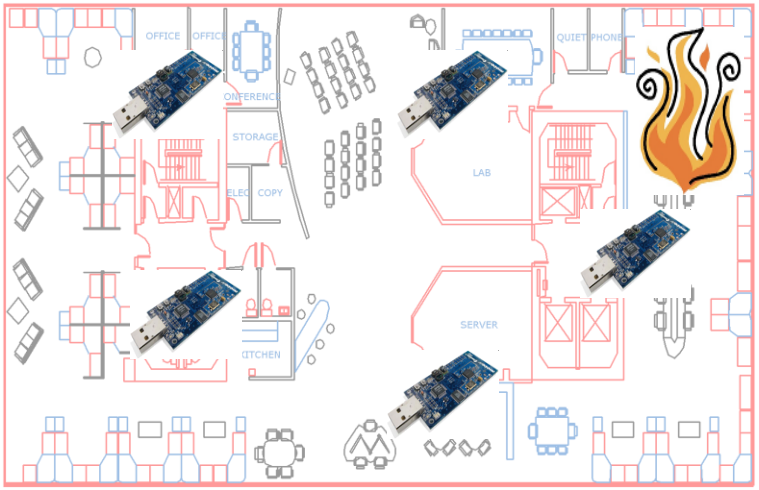


- ground set $\mathcal{V}$
- (scoring) function

$$F : 2^{\mathcal{V}} \rightarrow \mathbb{R}_+$$

$$\max \quad F(S)$$

Informative Subsets

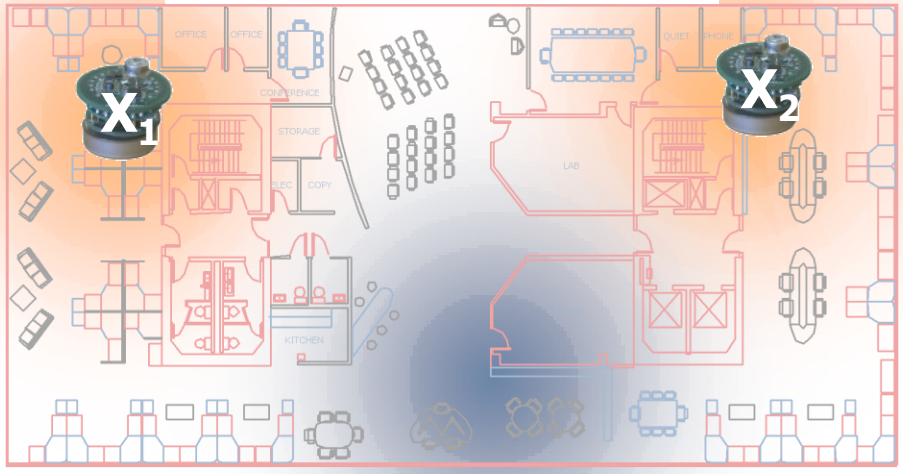


- where put sensors?
- which experiments?
- summarization

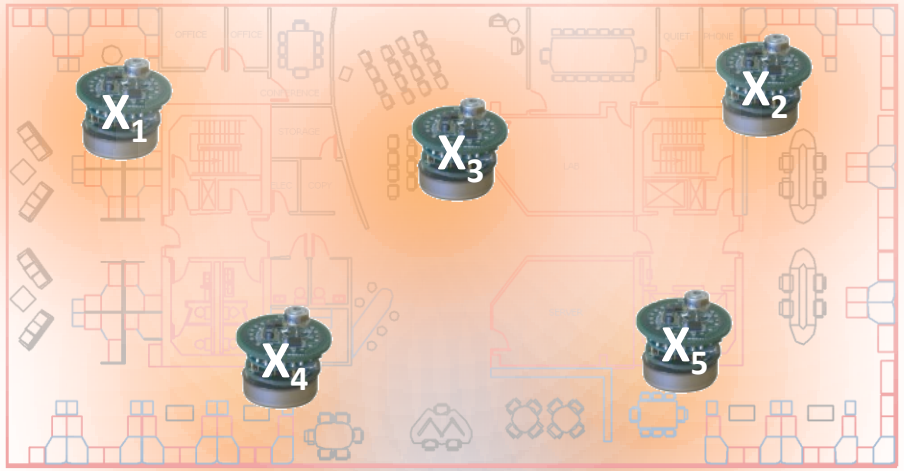
$$F(S) = \text{“information”}$$

Diminishing marginal gains

placement A = {1,2}



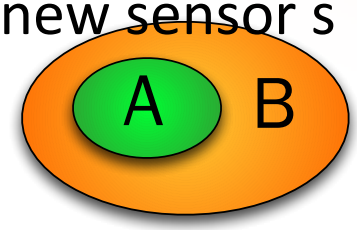
placement B = {1,…,5}



Big gain

new sensor s

small gain



$$A \subseteq B$$

$$F(A \cup s) - F(A) \quad \geq \quad F(B \cup s) - F(B)$$

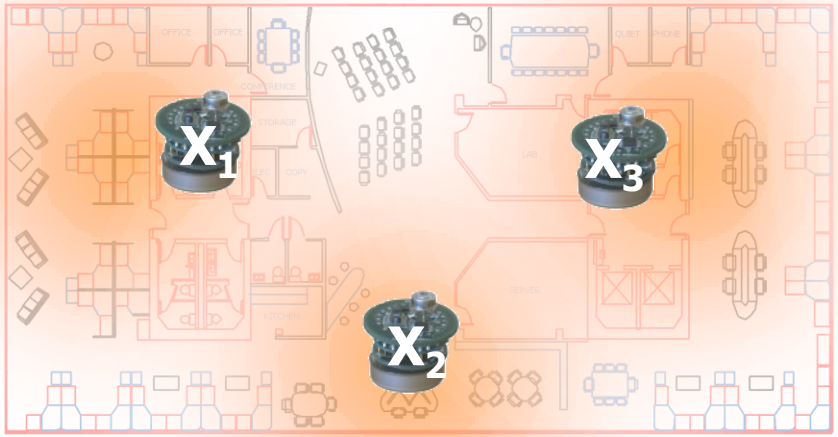
Sensor placement

Utility of having sensors at subset A of all locations

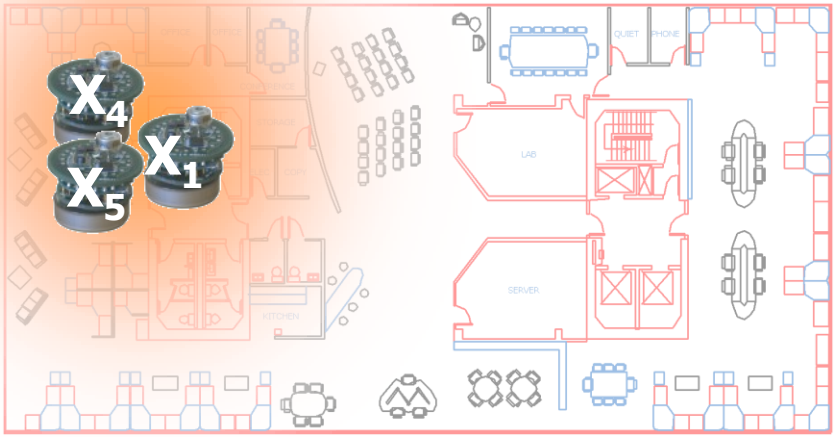
$$F(A) = H(\mathbf{Y}) - H(\mathbf{Y} \mid \mathbf{X}_A)$$

Uncertainty about temperature Y **before** sensing

Uncertainty about temperature Y **after** sensing



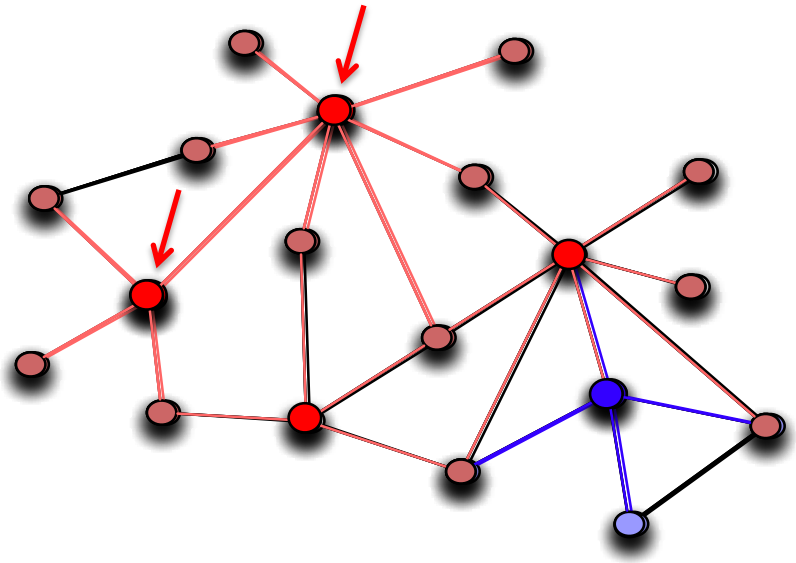
A={1,2,3}: High value F(A)



A={1,4,5}: Low value F(A)

Maximizing Influence

$$F(S) = \text{expected } \# \text{ infected nodes}$$



$$F(S \cup s) - F(S) \quad \geq \quad F(T \cup s) - F(T)$$

Kempe, Kleinberg & Tardos 2003

Summarization

- videos, text, pictures ...
- would like:
  relevance, reliability, diversity

Summarization

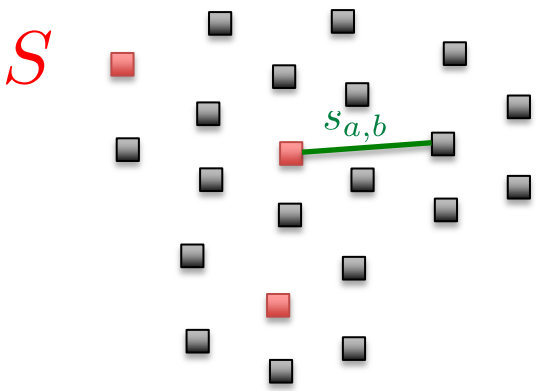
$$F(S) = R(S) + D(S)$$

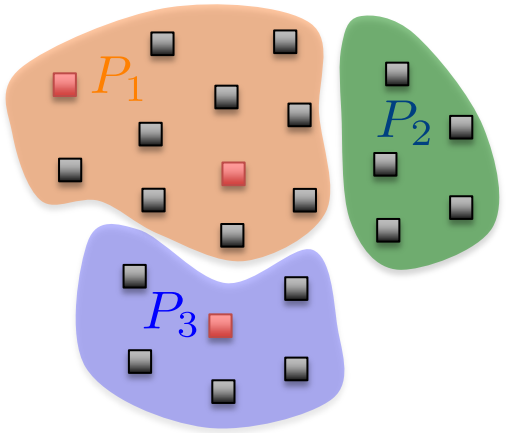
- Coverage / relevance

$$R(S) = \sum_{a \in \mathcal{V}} \max_{b \in S} s_{a,b}$$

- Diversity

$$D(S) = \sum_{j=1}^{m} \sqrt{|S \cap P_j|}$$



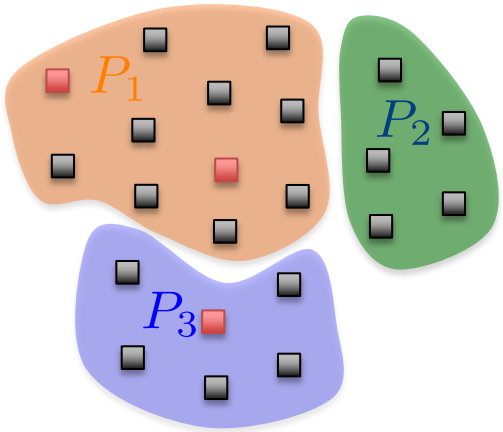


(Simon et al 2007, Lin & Bilmes 2011&2012, Tschiatschek et al 2014, Kim et al 2014, Gygli et al 2015, ...)

Diversity

- Diversity

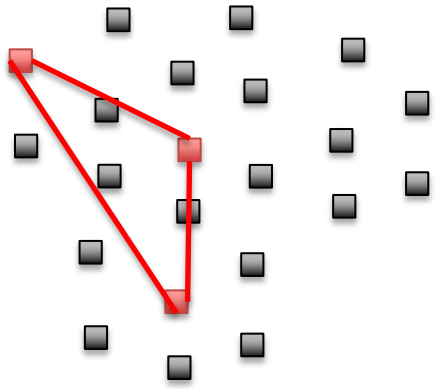
$$D(S) = \sum_{j=1}^{m} \sqrt{|S \cap P_j|}$$



increasing

Another diversity function ...

$$D(S) = -\sum_{a,b \in S} s_{a,b}$$



decreasing

Summarization: results

	R	F
$\mathcal{L}_1(S) + \lambda \mathcal{R}_Q(S)$	12.18	12.13
$\mathcal{L}_1(S) + \sum_{\kappa=1}^{3} \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$	**12.38**	**12.33**
Toutanova et al. (2007)	11.89	11.89
Haghighi and Vanderwende (2009)	11.80	-
Celikyilmaz and Hakkani-tür (2010)	11.40	-
Best system in DUC-07 (peer 15), using web search	**12.45**	12.29

(Lin & Bilmes 2011)

Many more functions are possible …

➔Learn a weighted combination: structured prediction

➔works even better!

see also papers in this CVPR: Xu et al., Gygli et al.,

(Lin & Bilmes 2012, Tschiatschek et al 2014, Gygli et al 2015, …)

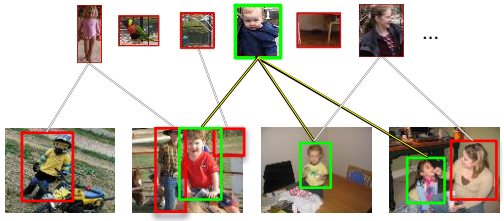
More maximization ...

co-segmentation
by maximizing
anisotropic diffusion
(Kim et al 2011)

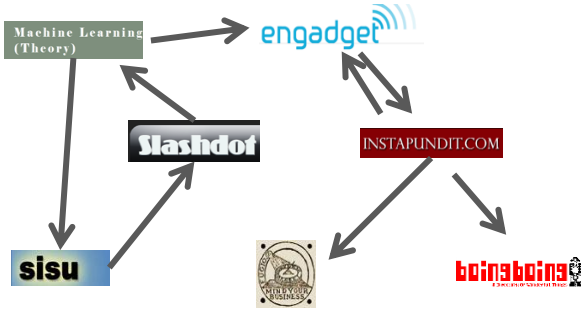
environmental monitoring
(Krause, ...)

$$\max \quad F(S)$$

weakly supervised
object detection
(Song et al 2014)



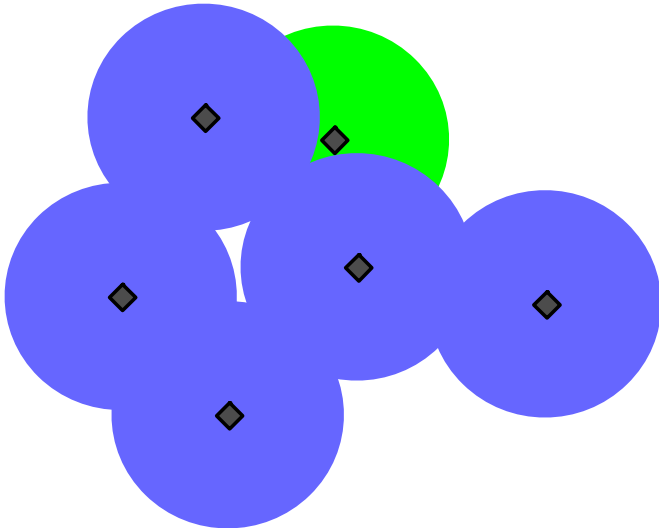
diverse
recommendations
(Yue & Guestrin)

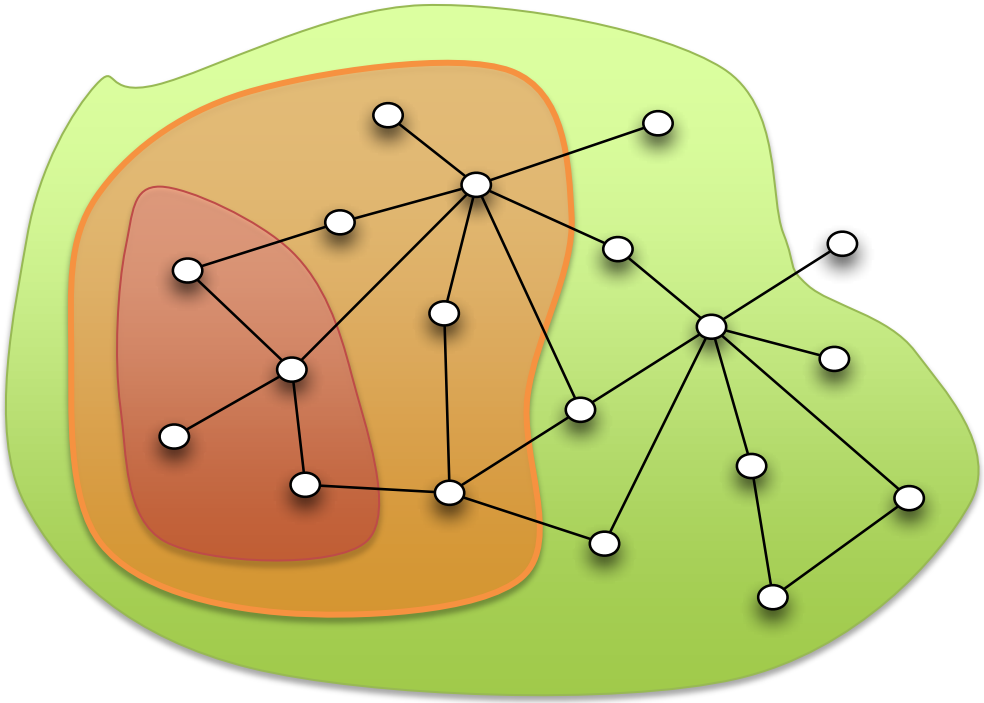


inferring networks
(Gomez Rodriguez et al 2012)

Monotonicity

$$\text{if} \quad S \subseteq T \quad \text{then} \quad F(S) \leq F(T)$$





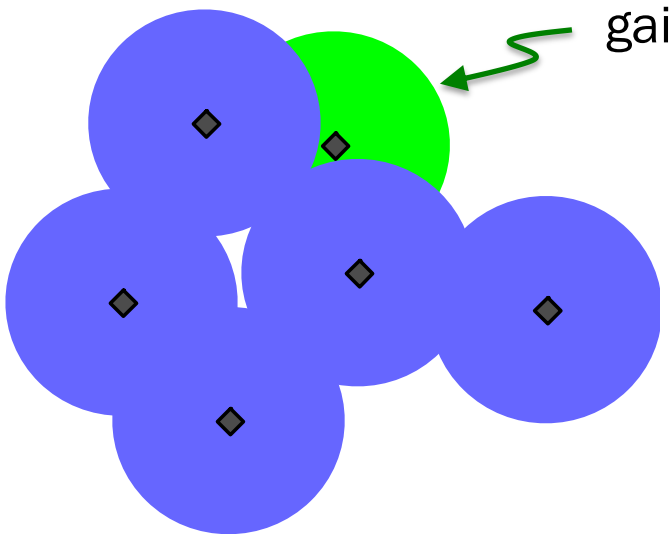
3 5 1

Monotonicity – how check?

$$\text{if } A \subseteq B \quad \text{then} \quad F(A) \leq F(B)$$

Let $B = A \cup \{a\}$.

$$\underbrace{F(A \cup \{a\}) - F(A)}_{\text{marginal gain}} \geq 0.$$



$$F(A) = \left| \bigcup_{a \in A} \text{area}(a) \right| - \sum_{a \in A} c(a)$$

Maximizing monotone functions

$$\text{if } A \subseteq B \quad \text{then} \quad F(A) \leq F(B)$$

$$\max \; F(S)$$

- NP-hard
- approximation: greedy algorithms

Concave aspects

- submodularity:

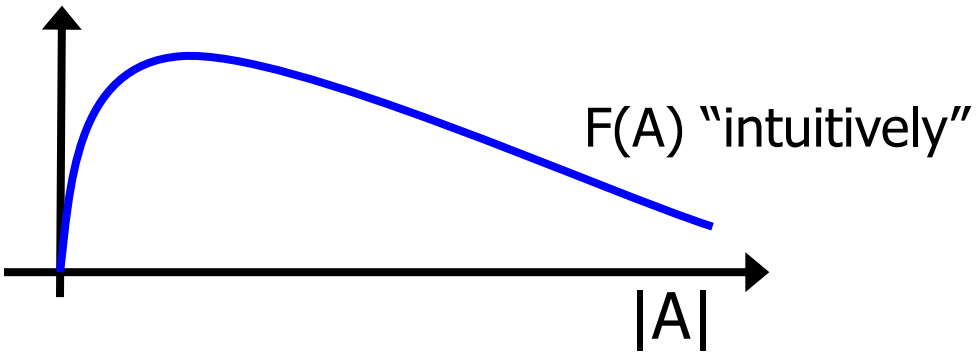
$$A \subseteq B, \;\; s \notin B :$$

$$F(A \cup s) - F(A) \quad \geq \quad F(B \cup s) - F(B)$$

- concavity:

$$a \leq b, \;\; s > 0 :$$

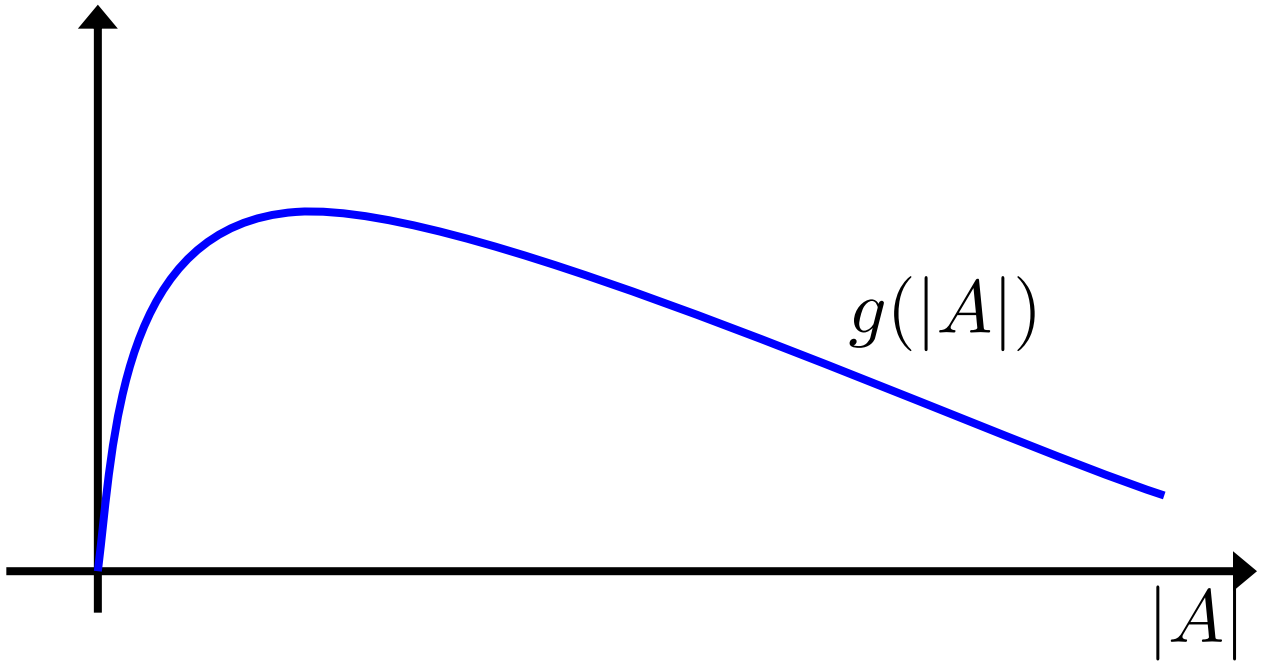
$$f(a + s) - f(a) \quad \geq \quad f(b + s) - f(b)$$



Submodularity and concavity

- suppose $g : \mathbb{N} \to \mathbb{R}$ and $F(A) = g(|A|)$

$F(A)$ submodular *if and only if* ...g is concave



Maximizing monotone functions

$$\max_S \; F(S) \;\; \text{s.t.} \;\; |S| \leq k$$

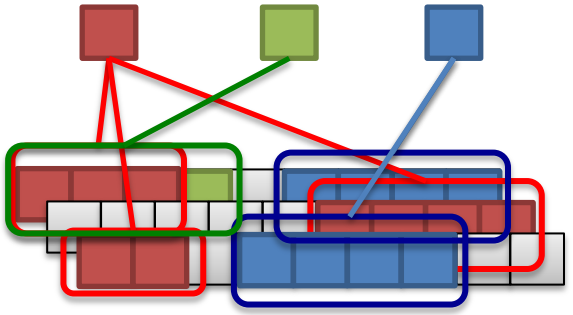
- greedy algorithm:

$S_0 = \emptyset$

for *i = 0, …, k-1*

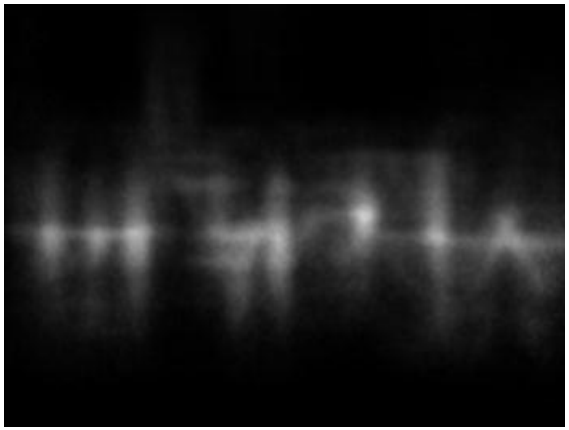
$$e^* = \arg\max_{e \in \mathcal{V} \setminus S_i} F(S_i \cup \{e\})$$

$$S_{i+1} = S_i \cup \{e^*\}$$

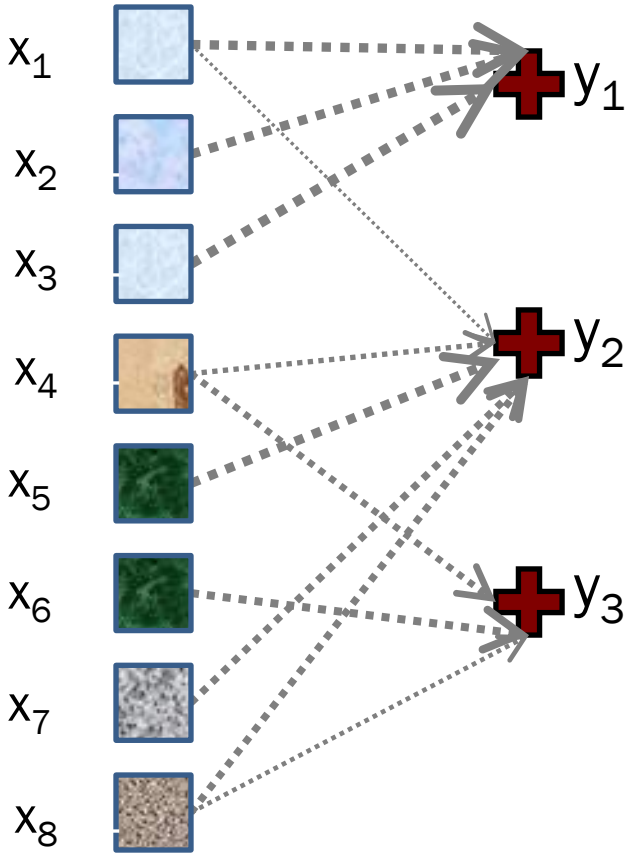


How "good" is S_k ?

Pedestrian detection



x_j = index of hypothesis explaining x_j



Voting elements

Hypotheses

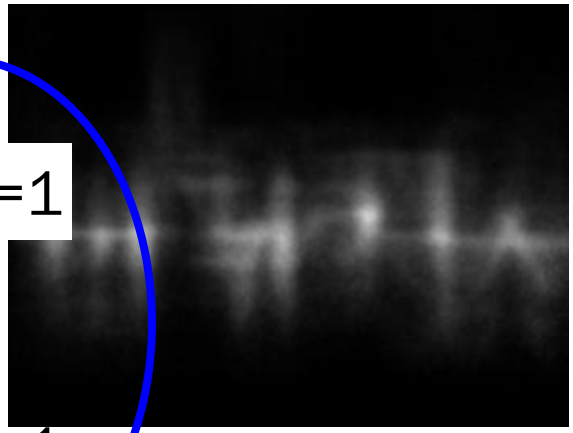
y_i = 1: object i present

y_i = 0: object i not present

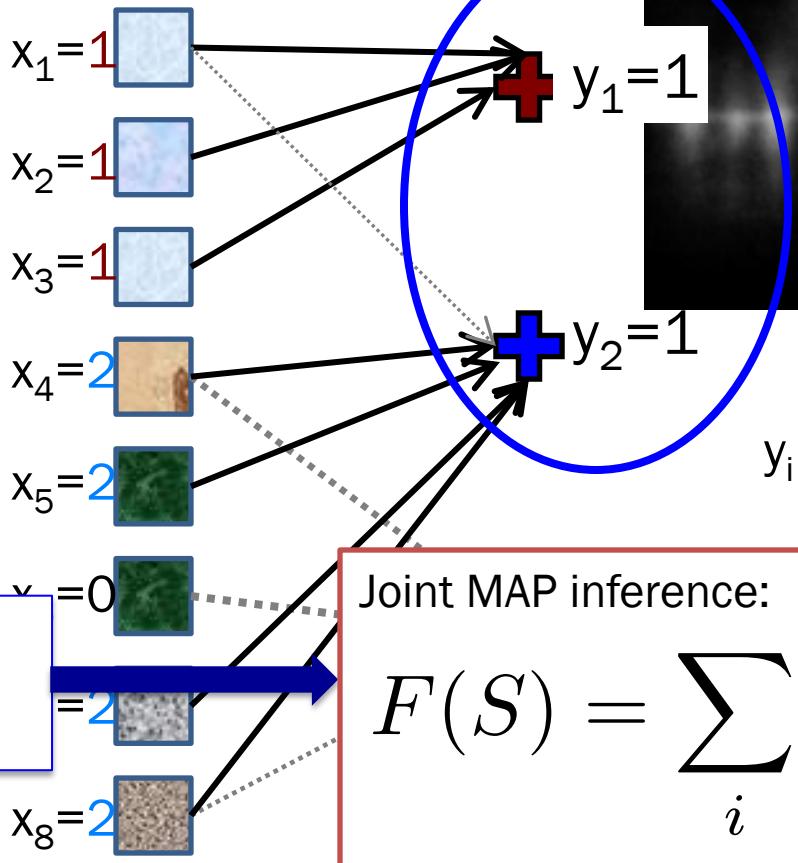
Illustrations courtesy of Pushmeet Kohli

(Barinova et al.'10)

Object detection



x_j = index of hypothesis explaining x_j



y_i = 1: object i present

submodular maximization!

Joint MAP inference:

$$F(S) = \sum_i \max_{j \in S} w_{i,j}$$

Weight element i wrt hyp. j

Voting elements

Illustrations courtesy of Pushmeet Kohli

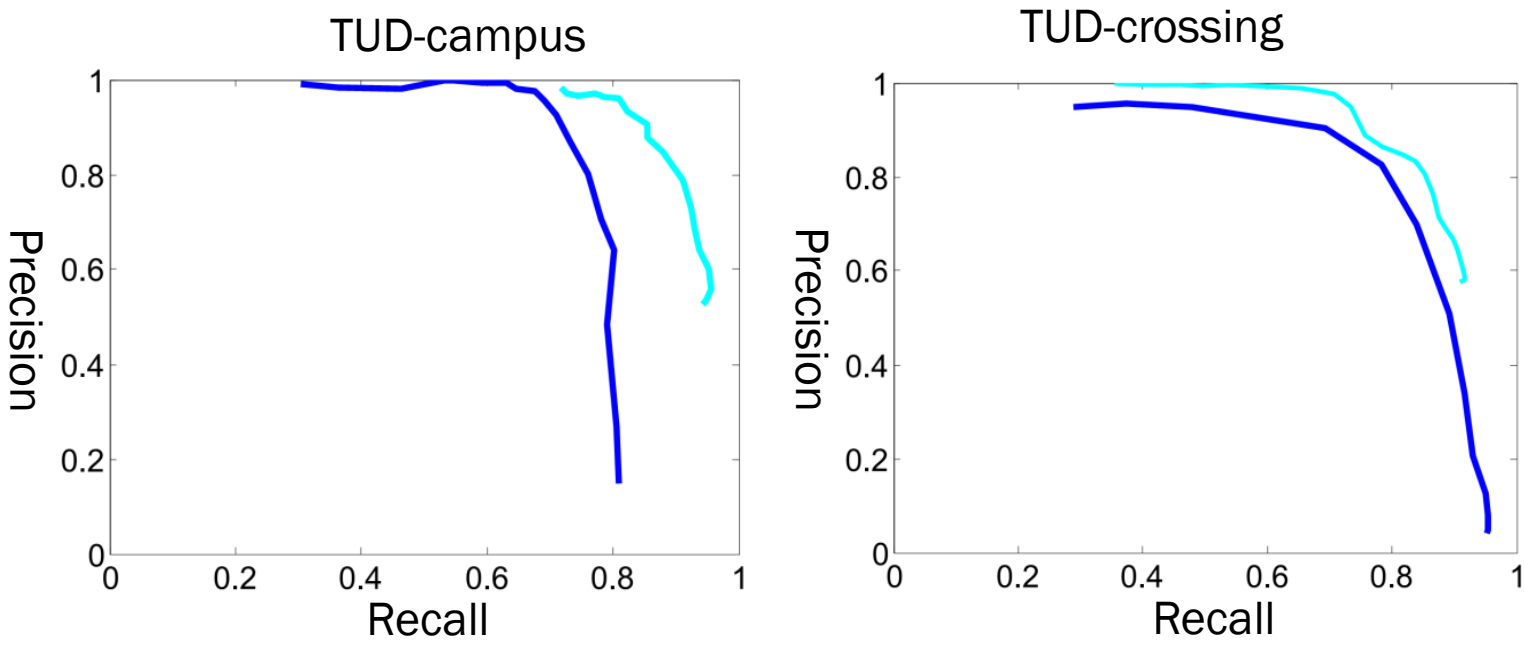
Inference

Datasets from [Andriluka et al. CVPR 2008]
(with strongly occluded pedestrians added)

Using the Hough forest trained in [Gall&Lempitsky CVPR09]

Illustrations courtesy of Pushmeet Kohli

Results for pedestrians detection

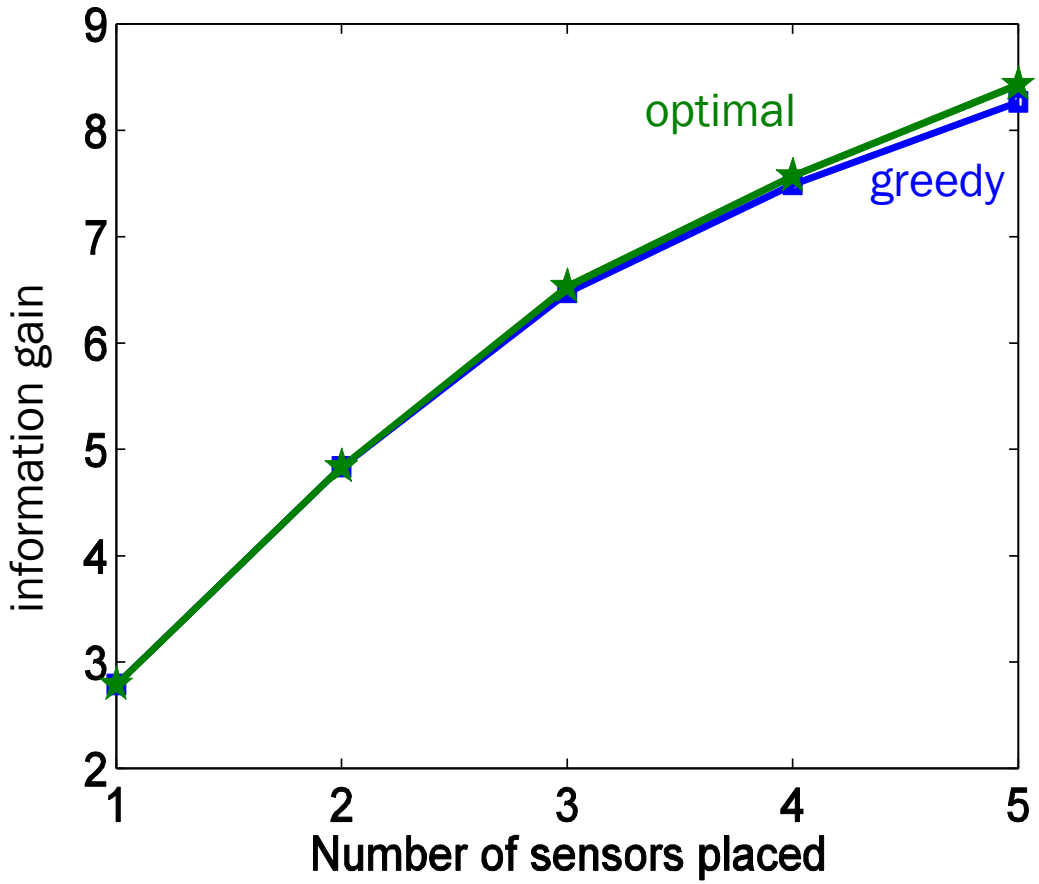


Blue = Hough transform + non-maximum suppression
Light-blue = greedy detection

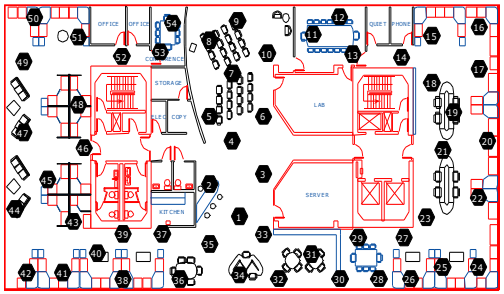
(submodularity for detection also in (Blaschko'11)

How good is greedy? in practice…

empirically:



sensor placement



How good is greedy? in theory...

$$\max_S \quad F(S) \quad \text{s.t.} \quad |S| \leq k$$

Theorem *(Nemhauser, Fisher, Wolsey `78)*

F monotone submodular, S_k solution of greedy. Then

$$F(S_k) \quad \geq \quad \left(1 - \frac{1}{e}\right) F(S^*)$$

optimal solution

in general, no poly-time algorithm can do better than that!

Questions

- What if I have more complex constraints?
  - matroid constraints: later (Sri)
  - budget constraints

- Greedy takes $O(nk)$ time. What if n, k are large?

- What if my function is not monotone?

More complex constraints: budget

$$\max \quad F(S) \quad \text{s.t.} \quad \sum_{e \in S} c(e) \leq B$$

1. run greedy: S_{gr}
2. run a modified greedy: S_{mod}

$$e^* = \arg\max \frac{F(S_i \cup \{e\}) - F(S_i)}{c(e)}$$

3. pick better of S_{gr} , S_{mod}

➔ approximation factor: $\frac{1}{2}\left(1 - \frac{1}{e}\right)$

even better but less fast: partial enumeration *(Sviridenko, 2004)* or filtering *(Badanidiyuru & Vondrák 2014)*

(Leskovec et al 2007)

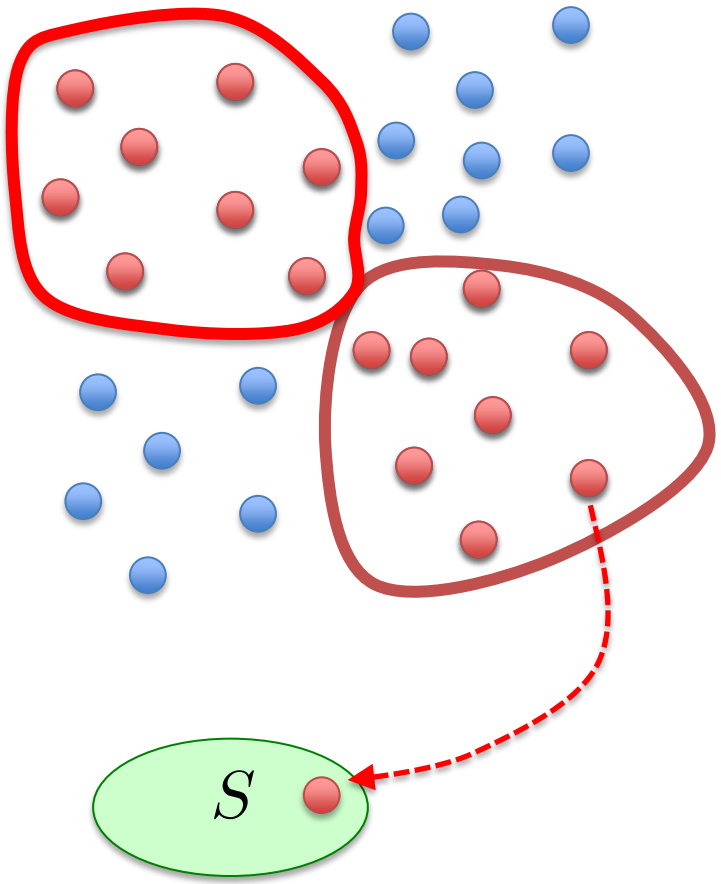
Questions

- What if I have more complex constraints?
  - matroid constraints: later (Sri)
  - budget constraints

- Greedy takes $O(nk)$ time. What if n, k are large?
  - faster sequential algorithms
  - parallel / distributed
  - exploit structure of V

- What if my function is not monotone?

Making greedy faster: stochastic



$$\max_S \; F(S) \;\; \text{s.t.} \;\; |S| \leq k$$

for i=1…k:

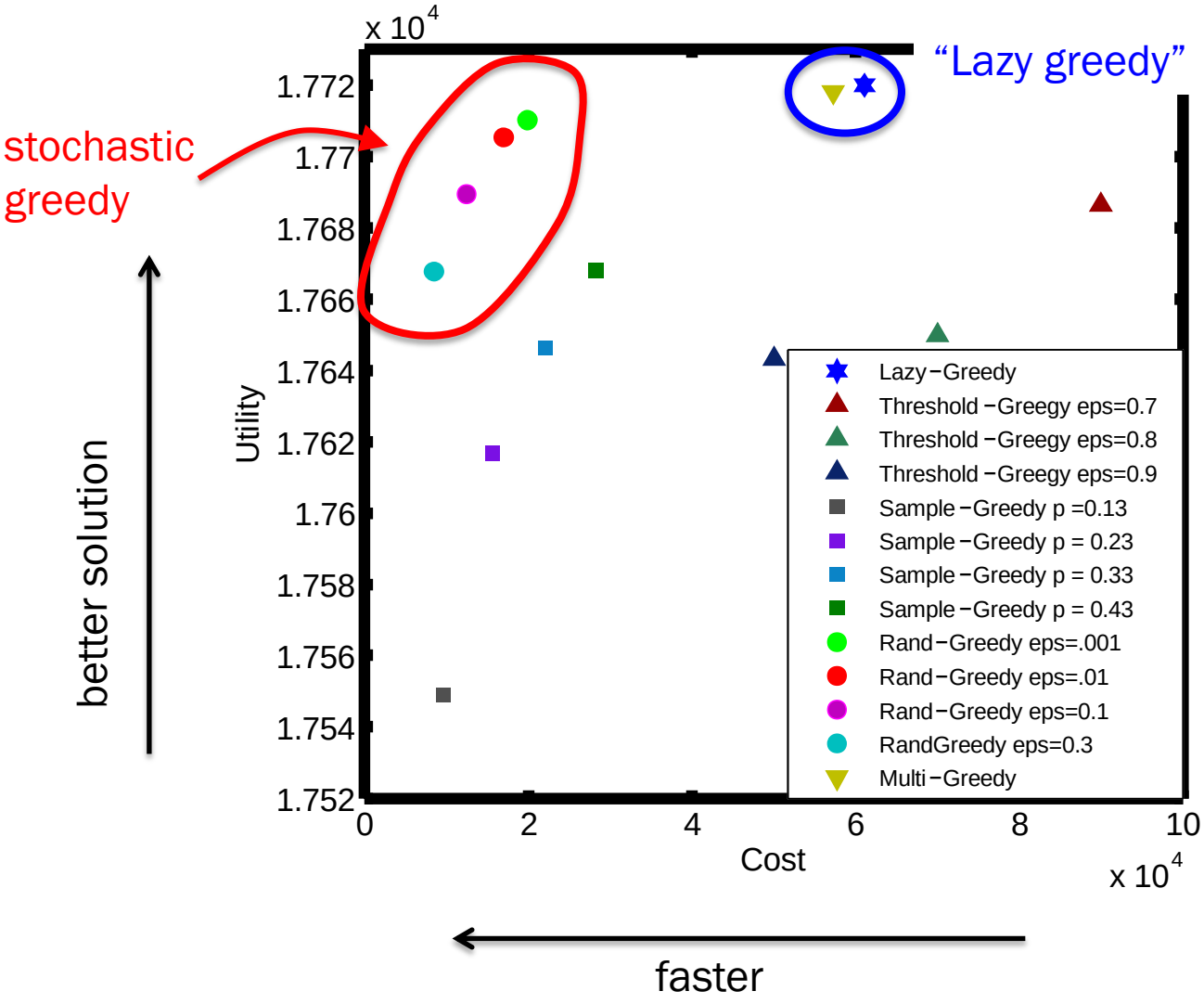
- randomly pick set *T* of size $\frac{n}{k} \log \frac{1}{\epsilon}$
- find best *a* element in *T* and add

$$a_i = \arg\max_{a \in T} F(a|S_{i-1})$$

$$S_i \leftarrow S_{i-1} \cup \{a_i\}$$

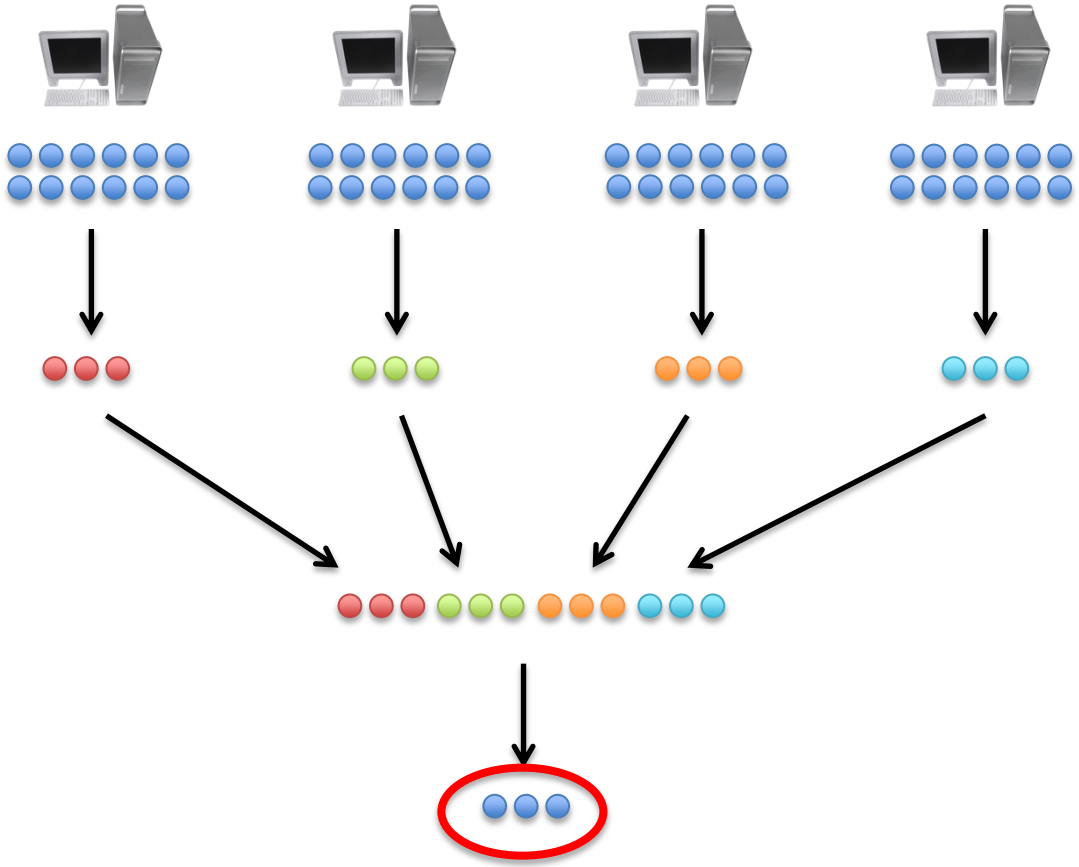
(Mirzasoleiman et al 2014)

Performance




even more data ...
distributed greedy algorithm?

Distributed greedy algorithms



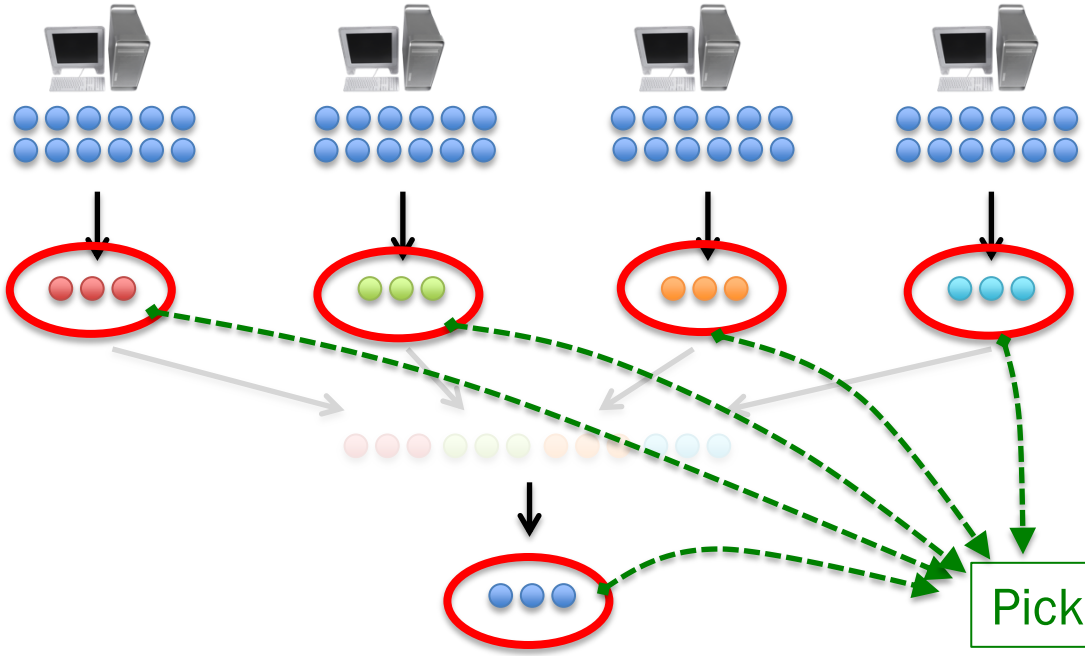
greedy is sequential.
pick in parallel??

pick *k* elements
on each machine.

combine and run
greedy again.

Is this useful?

Distributed greedy algorithms



pick in parallel
from *m* machines

Is this useful?

Pick the best of m+1 solutions

Approximation factor:

$$\frac{1}{\min\{\sqrt{k}, m\}}$$

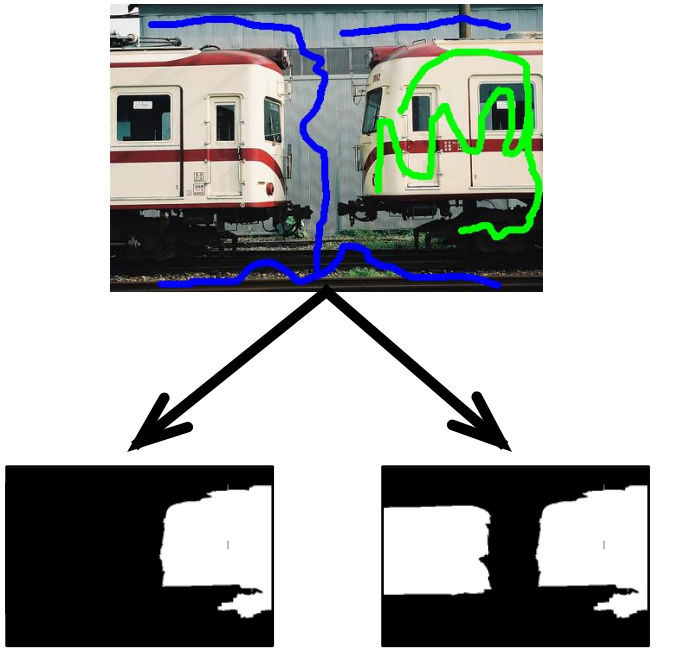
better with geometric structure

New approximation factor:

$$\frac{1}{2}\left(1 - \frac{1}{e}\right)$$

(Mirzasoleiman et al 2013, de Ponte Barbosa et al 2015)

Even larger …



One object? Two objects?

Find a set of *k* good but different solutions

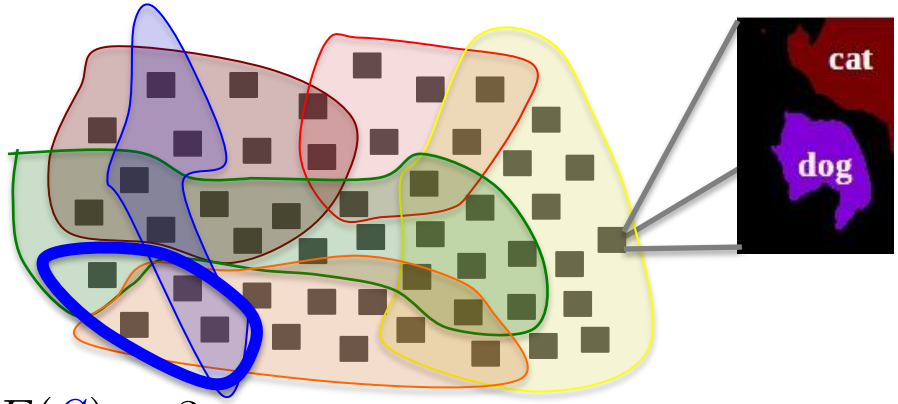
➔ diversity via submodular function

$\mathcal{Y}$ = set of all possible solutions (labelings)

exponentially large! ☹

can we run a greedy algorithm? for some submodular function?

Greedy in structured output spaces



$F(S) = 3$

$$e^* = \arg\max_{e \in \mathcal{V} \setminus S_i} F(S_i \cup \{e\}) \; *$$

divide space into groups

$F(S) = \#\text{groups intersecting } S$

argmax $*$ solved via HOP inference!

labels used

HOP: label costs
(Delong et al 2010)

label transitions

HOP: cooperative cuts
(Jegelka & Bilmes 2011)

Hamming balls

HOP: cardinality potentials
(Tarlow et al 2010)

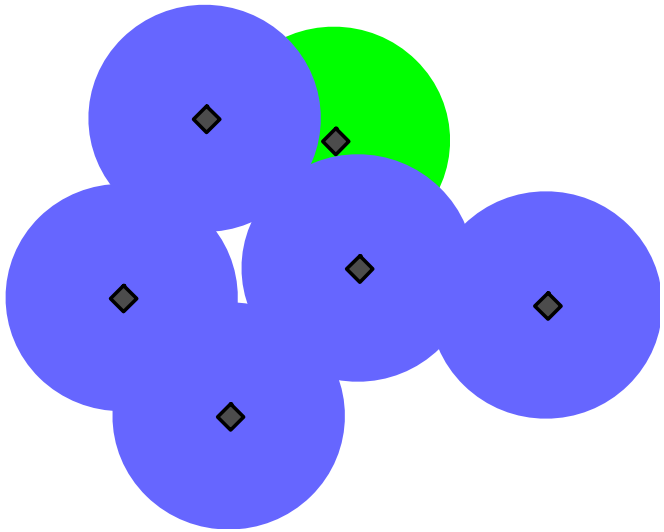
(Prasad et al 2014)

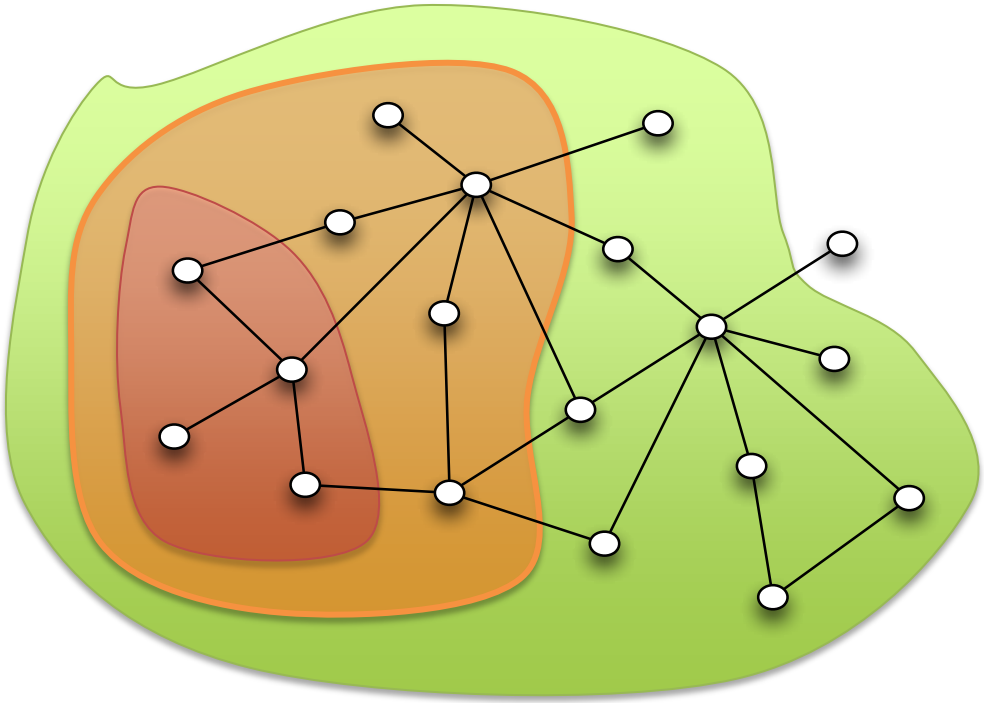
Questions

- What if I have more complex constraints?
  - matroid constraints: later (Sri)
  - budget constraints
- Greedy takes *O(nk)* time. What if n, k are large?
  - stochastic
  - distributed
  - structured
- What if my function is not monotone?

Non-monotone functions

~~if $S \subseteq T$ then $F(S) \leq F(T)$~~



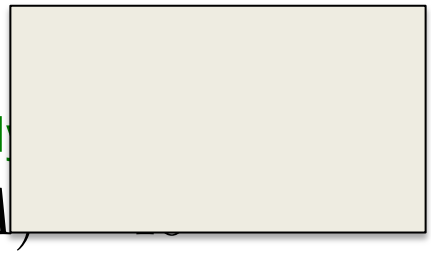


still assume:

$F(S) \geq 0 \quad \text{for all } S$

3 5 1

Greedy can fail ...



greedy

$F(A)$

$$F(A) = \left| \bigcup_{a \in A} \text{area}(a) \right| - \sum_{a \in A} c(a)$$

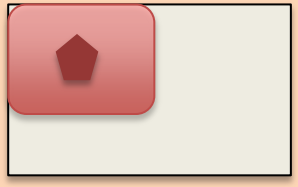
optimal solution

$F(A) = 95$

sensor 1

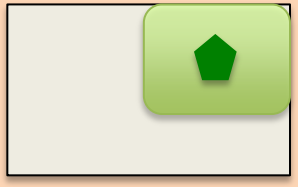
coverage: 100
cost: -60
gain 40

sensor 2



coverage: 30
cost: - 1
gain 29

sensor 3



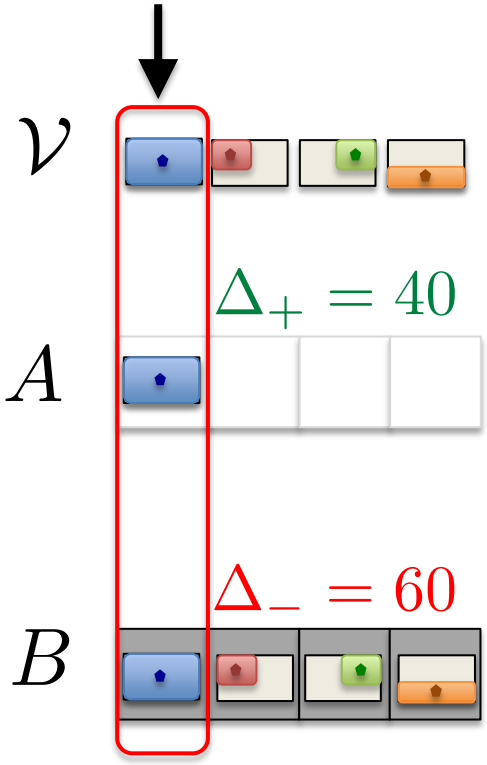
coverage: 30
cost: - 1
gain 29

sensor 4

coverage: 40
cost: - 3
gain 37

$$S_0 = \emptyset \qquad S_1 = \emptyset \cup \arg\max_{a \in \mathcal{V}} F(a)$$

Double (bidirectional) greedy



coverage: 100
cost: -60

Start: $A = \emptyset,\ B = \mathcal{V}$

for *i=1, …, n* *//add or remove?*

- gain of adding (to A):

$$\Delta_+ = [\, F(A \cup a_i) - F(A) \,]_+$$

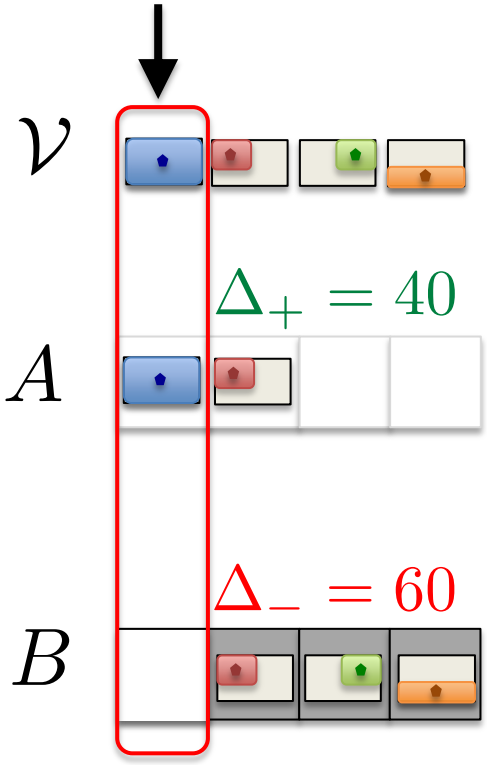
- gain of removing (from B):

$$\Delta_- = [\, F(B \setminus a) - F(B) \,]_+$$

add with probability

$$\mathbb{P}(\text{add}) = \frac{\Delta_+}{\Delta_+ + \Delta_-} \quad = 40\%$$

Double (bidirectional) greedy



Start: $A = \emptyset,\ B = \mathcal{V}$

for *i=1, …, n* *//add or remove?*

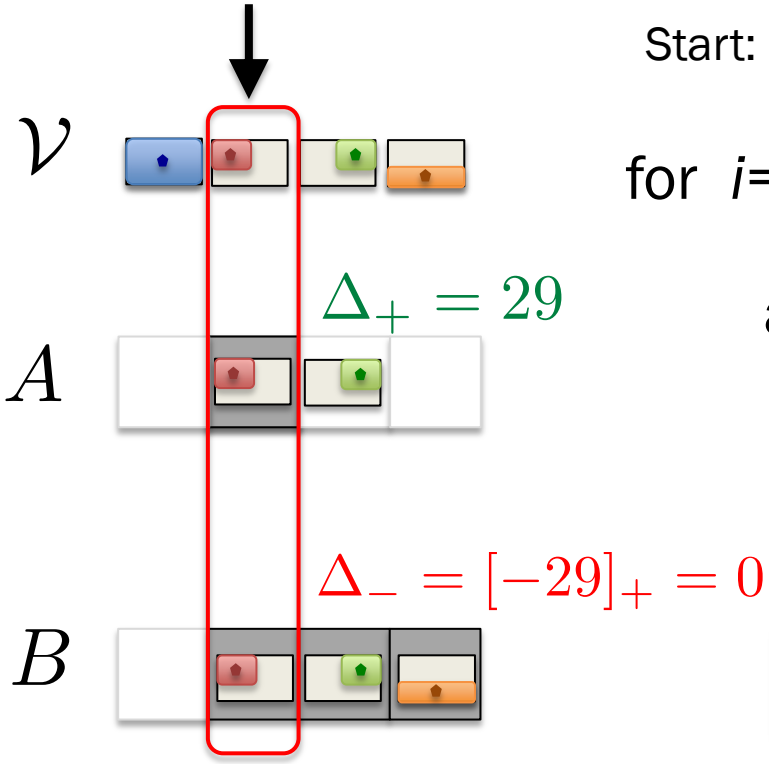
add with probability

$$\mathbb{P}(\text{add}) = \frac{\Delta_+}{\Delta_+ + \Delta_-}$$

add to A or remove from B

coverage: 100
cost: -60

Double (bidirectional) greedy



Start: $A = \emptyset,\ B = \mathcal{V}$

for *i=1, ..., n* *//add or remove?*

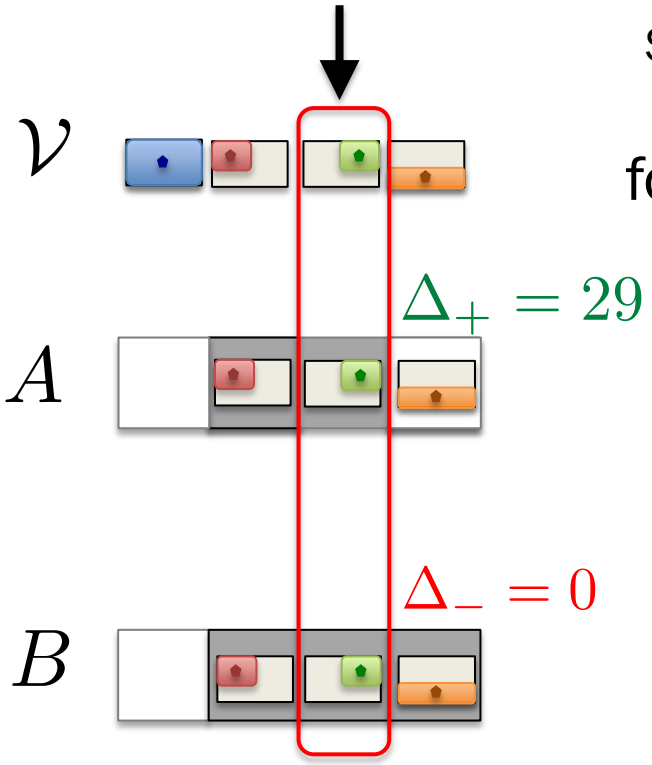
add with probability

$$\mathbb{P}(\text{add}) = \frac{\Delta_+}{\Delta_+ + \Delta_-} = \frac{29}{29}$$

add to A or remove from B

coverage: 30
cost: - 1

Double (bidirectional) greedy



Start: $A = \emptyset,\ B = \mathcal{V}$

for *i=1, …, n* *//add or remove?*

add with probability

$$\mathbb{P}(\text{add}) = \frac{\Delta_+}{\Delta_+ + \Delta_-} = \frac{29}{29}$$

add to A or remove from B

coverage: 30
cost: - 1

Double greedy

$$\max_{S \subseteq \mathcal{V}} F(S)$$

Theorem *(Buchbinder, Feldman, Naor, Schwartz '12)*

F submodular, S_g solution of double greedy. Then

$$\mathbb{E}[F(S_g)] \geq \tfrac{1}{2} F(S^*)$$

optimal solution

Non-monotone maximization

- alternatives to double greedy?
  local search *(Feige et al 2007)*

- constraints?
  possible, but different algorithms

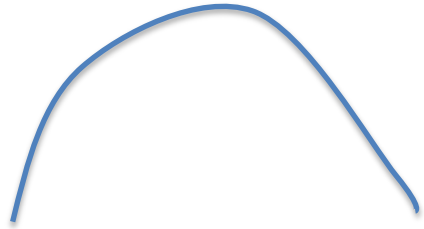
- distributed algorithms? yes!
  - divide-and-conquer as before *(de Ponte Barbosa et al 2015)*
  - concurrency control / Hogwild *(Pan et al 2014)*

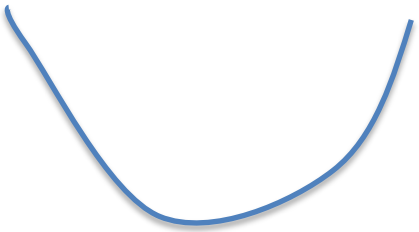
What we could not cover

- many more applications …
- probabilistic diversity models: determinantal point processes
- learning submodular functions
- adaptive submodularity

Submodular optimization

maximization:
concave aspects





minimization:
convex aspects

Submodular maximization: summary

- many applications: diverse, informative subsets

- NP-hard, but greedy or local search
- distinguish monotone / non-monotone

- several constraints possible
  (monotone and non-monotone)

- other possibility: multilinear extension