Beyond Convexity – Submodularity in Machine Learning

Andreas Krause, Carlos Guestrin

Carnegie Mellon University

International Conference on Machine Learning | July 5, 2008

Select Lab

Carnegie Mellon

Acknowledgements

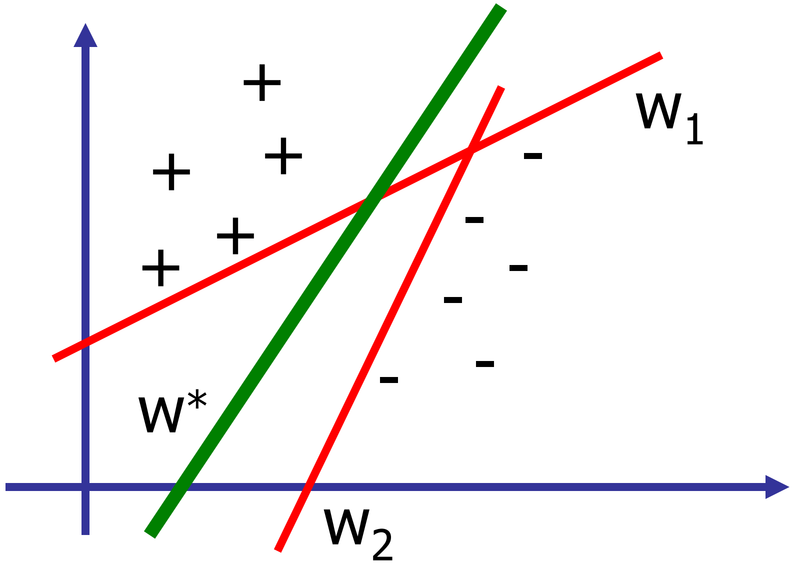
- Thanks for slides and material to Mukund Narasimhan, Jure Leskovec and Manuel Reyes Gomez

- MATLAB Toolbox and details for references available at

  http://www.submodularity.org

  Algorithms implemented → $\mathcal{M}$

Optimization in Machine Learning



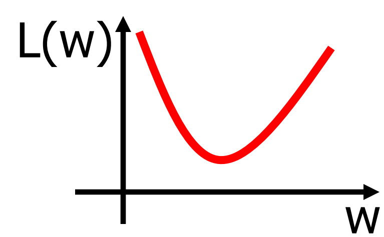
Classify + from – by finding a separating hyperplane (parameters w)

Which one should we choose?

Define loss L(w) = “1/size of margin”

➔ Solve for best vector $\boxed{w^* = \text{argmin}_w L(w)}$

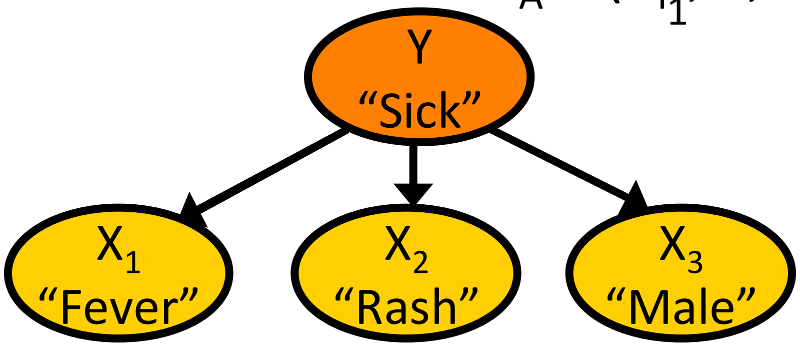
Key observation: Many problems in ML are convex!



➔ no local minima!! ☺

Feature selection

- Given random variables Y, $X_1, \ldots X_n$
- Want to predict Y from subset $X_A = (X_{i_1},\ldots,X_{i_k})$



Naïve Bayes Model

Want k most informative features:

$$A^* = \operatorname{argmax} IG(X_A; Y) \text{ s.t. } |A| \leq k$$

where $IG(X_A; Y) = H(Y) - H(Y \mid X_A)$

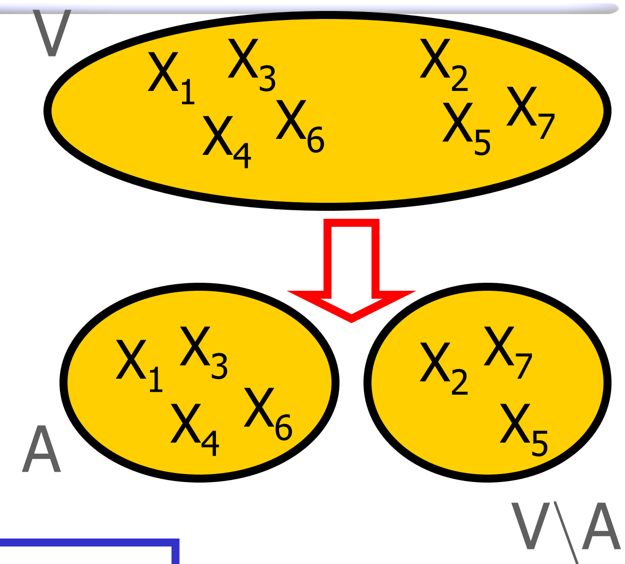
Uncertainty before knowing X_A

Uncertainty after knowing X_A

Problem inherently combinatorial!

Factoring distributions

- Given random variables $X_1,\ldots,X_n$
- Partition variables V into sets A and $V\setminus A$ as independent as possible



Formally: Want

$$\boxed{A^* = \operatorname{argmin}_A I(X_A; X_{V\setminus A}) \quad \text{s.t. } 0<|A|<n}$$

where $I(X_A,X_B) = H(X_B) - H(X_B \mid X_A)$

Fundamental building block in structure learning [Narasimhan&Bilmes, UAI '04]

Problem inherently combinatorial!

Combinatorial problems in ML

Given a (finite) set V, function F: $2^V \rightarrow R$, want

$$\boxed{A^* = \text{argmin } F(A) \quad \text{s.t. some constraints on } A}$$

Solving combinatorial problems:

- Mixed integer programming?

  Often difficult to scale to large problems

- Relaxations? (e.g., L1 regularization, etc.)

  Not clear when they work

- This talk:

  Fully combinatorial algorithms (spanning tree, matching, …)

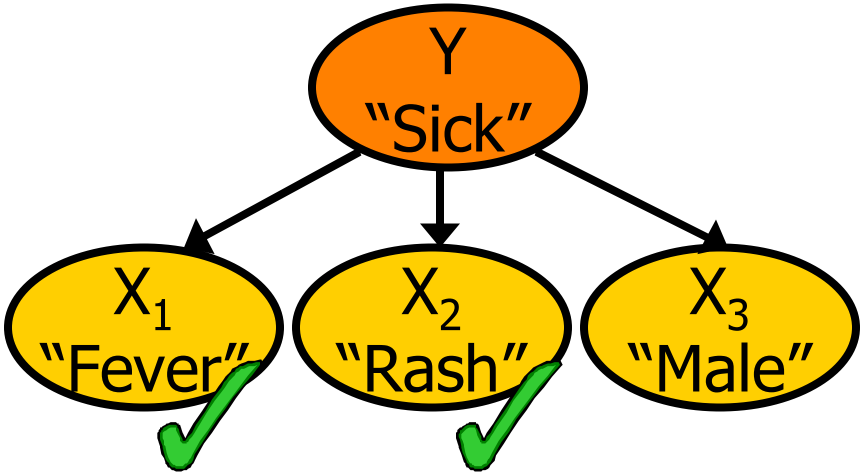
  Exploit problem structure to get guarantees about solution!

Example: Greedy algorithm for feature selection

- Given: finite set V of features, utility function $F(A) = IG(X_A; Y)$
- Want: $A^* \subseteq V$ such that

$$\mathcal{A}^* = \underset{|\mathcal{A}|\leq k}{\operatorname{argmax}} F(\mathcal{A})$$

NP-hard!



<u>Greedy algorithm:</u> M

Start with $A = \emptyset$

For i = 1 to k

$s^* := \operatorname{argmax}_s F(A \cup \{s\})$

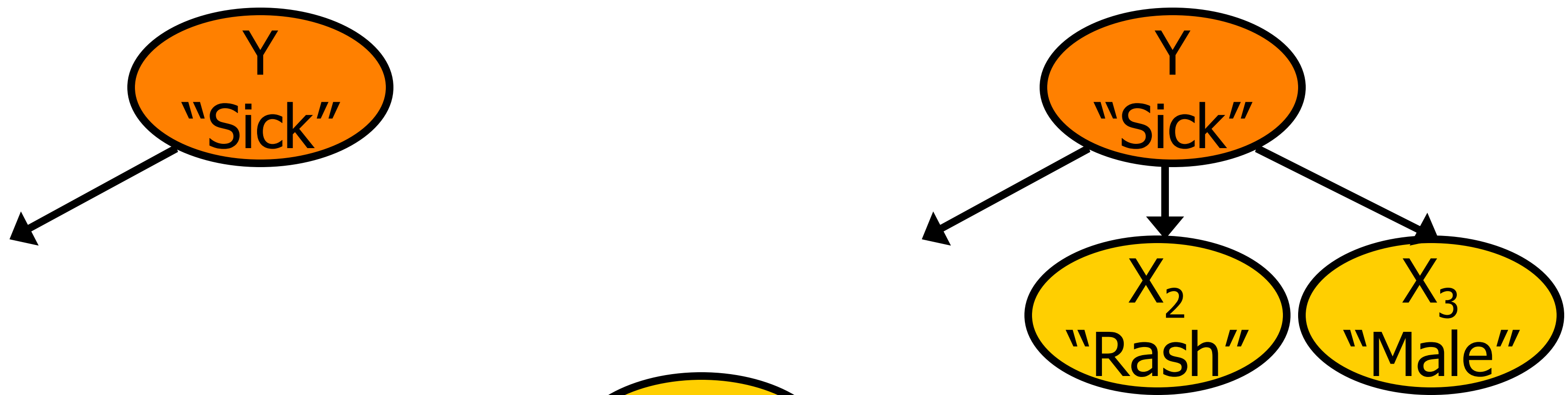
$A := A \cup \{s^*\}$

How well can this simple heuristic do?

Key property: Diminishing returns

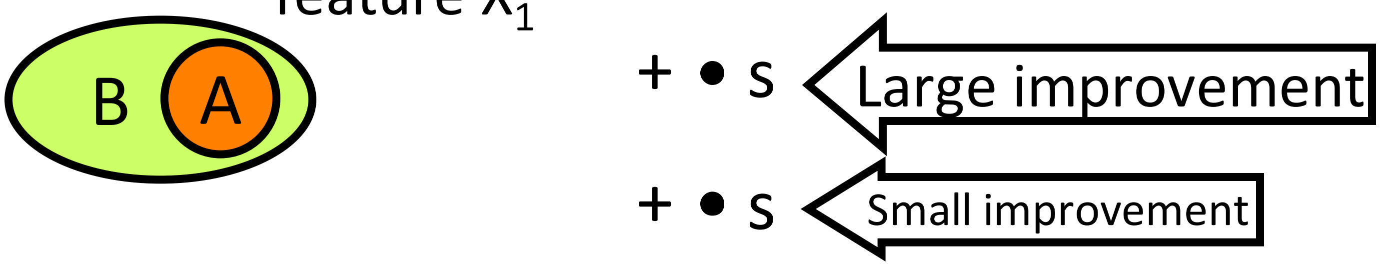
Selection A = {}

Selection B = {X_2,X_3}



Theorem [Krause, Guestrin UAI '05]**: Information gain F(A) in Naïve Bayes models is submodular!**

Submodularity:



For $A \subseteq B$, $F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B)$

Why is submodularity useful?

Theorem [Nemhauser et al '78]

Greedy maximization algorithm returns A_{greedy}:

$$F(A_{greedy}) \geq \underbrace{(1-1/e)}_{\sim 63\%} \max_{|A| \leq k} F(A)$$

- Greedy algorithm gives near-optimal solution!
- More details and exact statement later
- For info-gain: Guarantees best possible unless P = NP! [Krause, Guestrin UAI '05]

Submodularity in Machine Learning

- In this tutorial we will see that many ML problems are submodular, i.e., for F submodular require:
- Minimization: $A^* = \text{argmin } F(A)$
  - Structure learning ($A^* = \text{argmin } I(X_A; X_{V\setminus A})$)
  - Clustering
  - MAP inference in Markov Random Fields
  - …
- Maximization: $A^* = \text{argmax } F(A)$
  - Feature selection
  - Active learning
  - Ranking
  - …

Tutorial Overview

1. Examples and properties of submodular functions
2. Submodularity and convexity
3. Minimizing submodular functions
4. Maximizing submodular functions
5. Research directions, …

LOTS of applications to Machine Learning!!

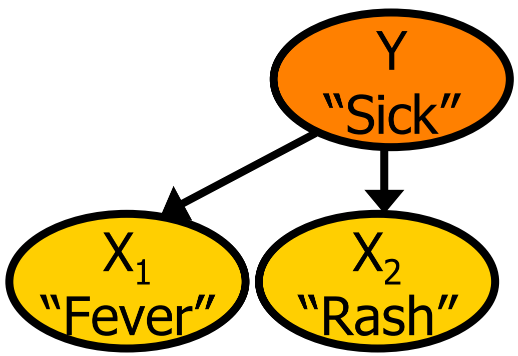
Submodularity

Properties and Examples

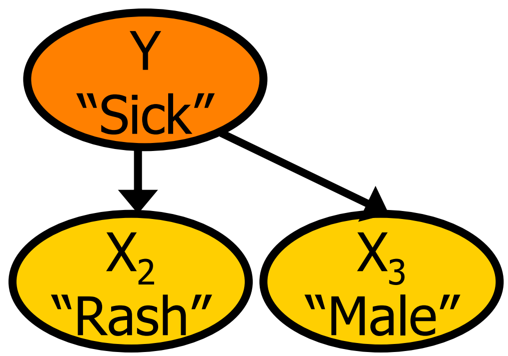
Set functions

- Finite set $V = \{1,2,\ldots,n\}$
- Function $F: 2^V \to R$
- Will always assume $F(\emptyset) = 0$ (w.l.o.g.)
- Assume black-box that can evaluate F for any input A
  - Approximate (noisy) evaluation of F is ok (e.g., [37])
- Example: $F(A) = IG(X_A; Y) = H(Y) - H(Y \mid X_A)$

$$= \sum_{y,x_A} P(x_A)\,[\log P(y \mid x_A) - \log P(y)]$$



$F(\{X_1,X_2\}) = 0.9$

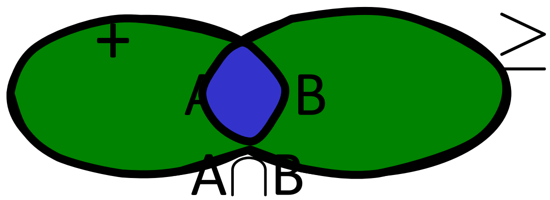


$F(\{X_2,X_3\}) = 0.5$

Submodular set functions

- Set function F on V is called submodular if

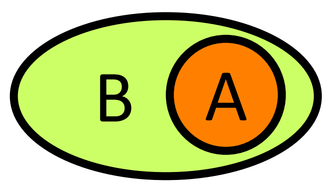
$$\text{For all } A,B \subseteq V: F(A)+F(B) \geq F(A\cup B)+F(A\cap B)$$



+

- Equivalent diminishing returns characterization:

Submodularity:



+ •S Large improvement

+ •S Small improvement

For $A\subseteq B$, $s \notin B$, $F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B)$

Submodularity and supermodularity

- Set function F on V is called submodular if

  1) For all $A,B \subseteq V$: $F(A)+F(B) \geq F(A\cup B)+F(A\cap B)$

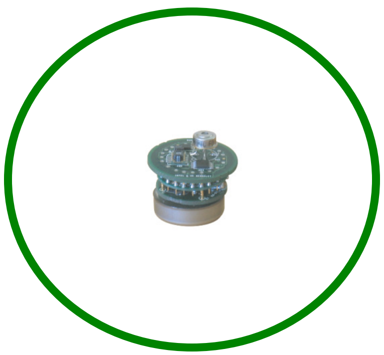
  $\Leftrightarrow$ 2) For all $A\subseteq B$, $s\notin B$, $F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B)$

- F is called supermodular if –F is submodular
- F is called modular if F is both sub- and supermodular

  for modular (“additive”) F, $F(A) = \sum_{i\in A} w(i)$

Example: Set cover

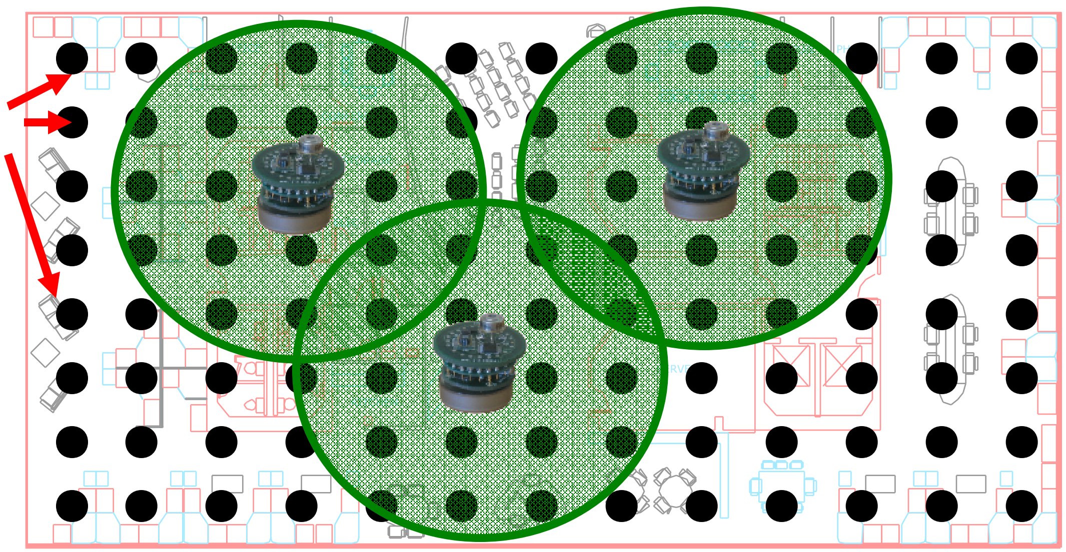
Place sensors
in building



Node predicts
values of positions
with some radius

Want to cover floorplan with discs

Possible
locations
V



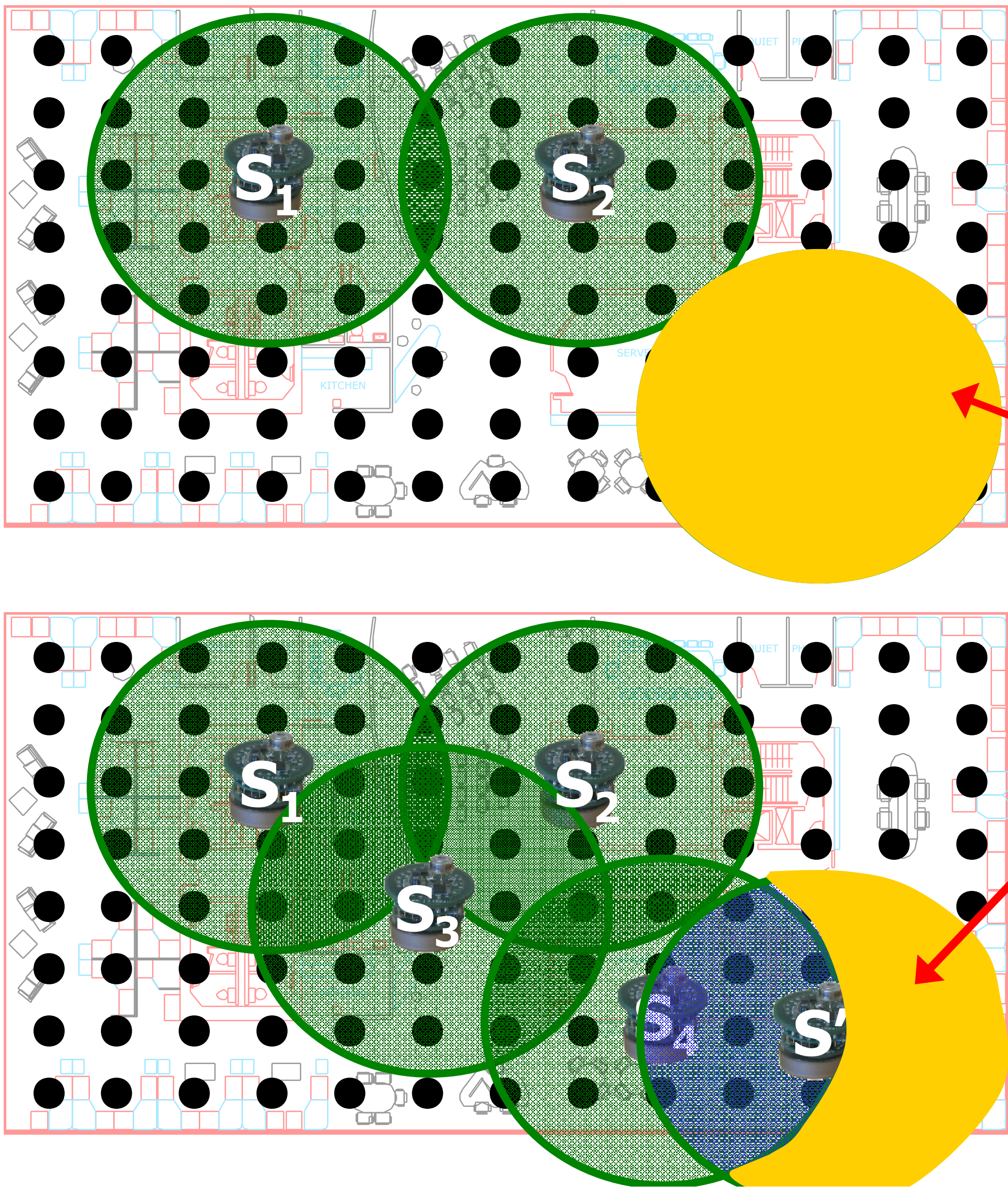
For $A \subseteq V$: $F(A)$ = "area
covered by sensors placed at A"

Formally:

W finite set, collection of n subsets $S_i \subseteq W$

For $A \subseteq V=\{1,\ldots,n\}$ define $F(A) = |\bigcup_{i \in A} S_i|$

Set cover is submodular



$A=\{S_1,S_2\}$

$F(A \cup \{S'\})-F(A)$

$\geq$

$F(B \cup \{S'\})-F(B)$

$B = \{S_1,S_2,S_3,S_4\}$

Example: Mutual information

- Given random variables $X_1,\ldots,X_n$
- $F(A) = I(X_A; X_{V\setminus A}) = H(X_{V\setminus A}) - H(X_{V\setminus A} \mid X_A)$

Lemma: Mutual information F(A) is submodular

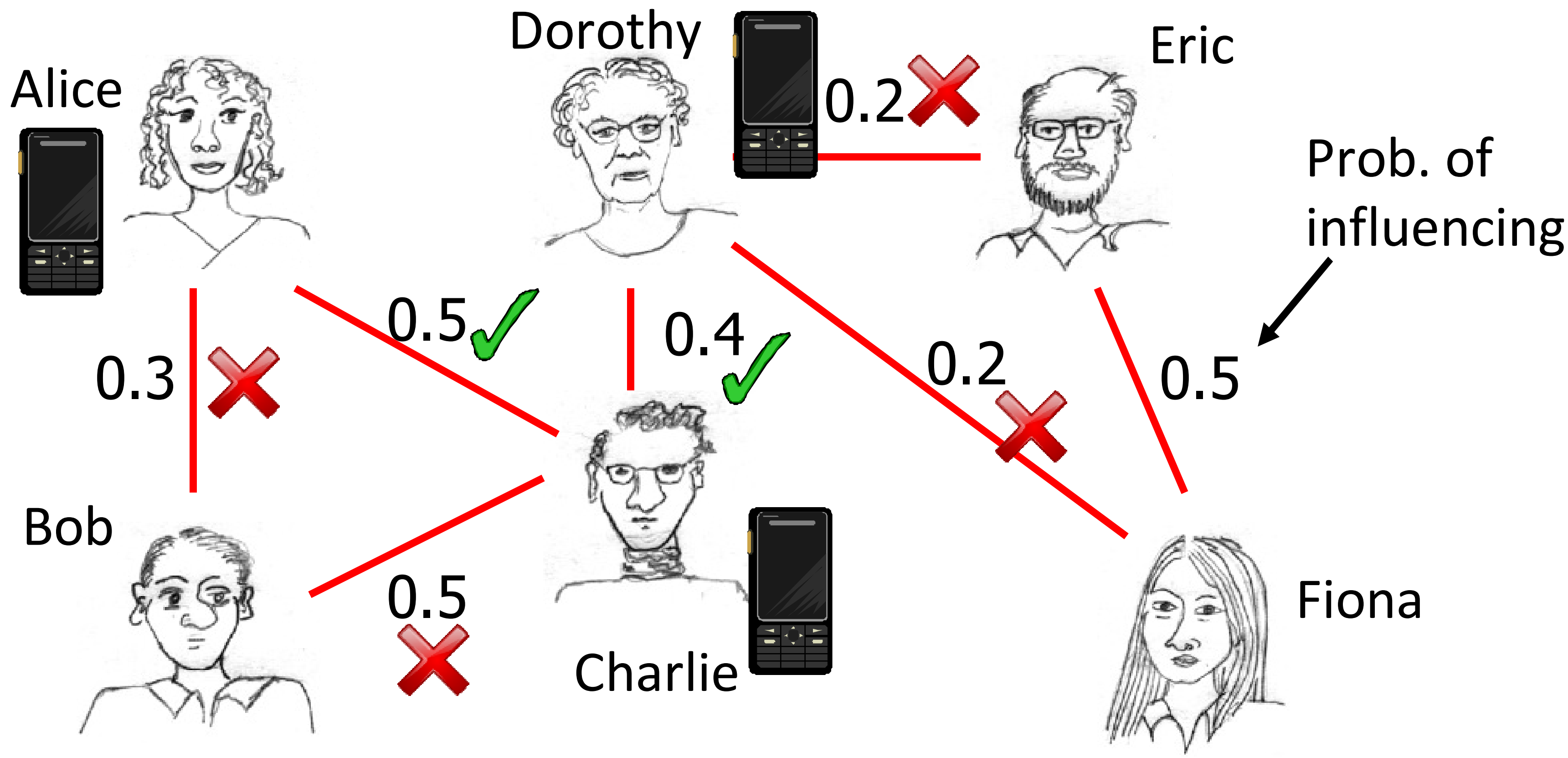
$$F(A \cup \{s\}) - F(A) = \underbrace{H(X_s \mid X_A)}_{\text{Nonincreasing in A:}} - \underbrace{H(X_s \mid X_{V\setminus(A\cup\{s\})})}_{\text{Nondecreasing in A}}$$

$A \subseteq B \Rightarrow H(X_s|X_A) \geq H(X_s|X_B)$

$\delta_s(A) = F(A\cup\{s\})-F(A)$ monotonically nonincreasing

$\Leftrightarrow$ F submodular ☺

Example: Influence in social networks [Kempe, Kleinberg, Tardos KDD '03]

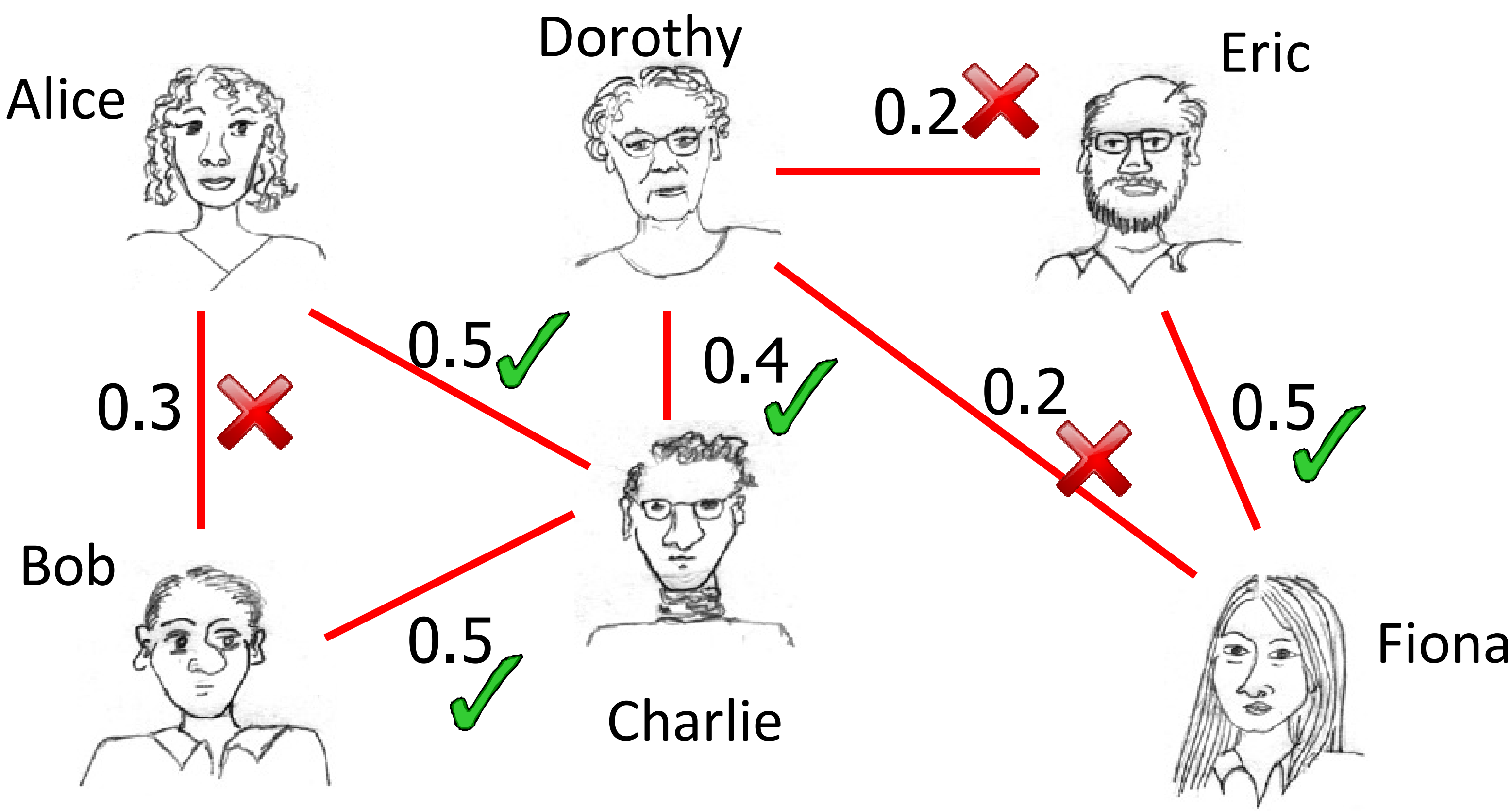


Who should get free cell phones?

V = {Alice,Bob,Charlie,Dorothy,Eric,Fiona}

F(A) = Expected number of people influenced when targeting A

Influence in social networks is submodular [Kempe, Kleinberg, Tardos KDD ’03]



Key idea: Flip coins **c** in advance ➔ “live” edges

$F_c(A)$ = People influenced under outcome **c** (set cover!)

$F(A) = \sum_c P(c) F_c(A)$ is submodular as well!

Closedness properties

$F_1,\ldots,F_m$ submodular functions on V and $\lambda_1,\ldots,\lambda_m > 0$

Then: $F(A) = \sum_i \lambda_i F_i(A)$ is submodular!

Submodularity closed under nonnegative linear combinations!

Extremely useful fact!!

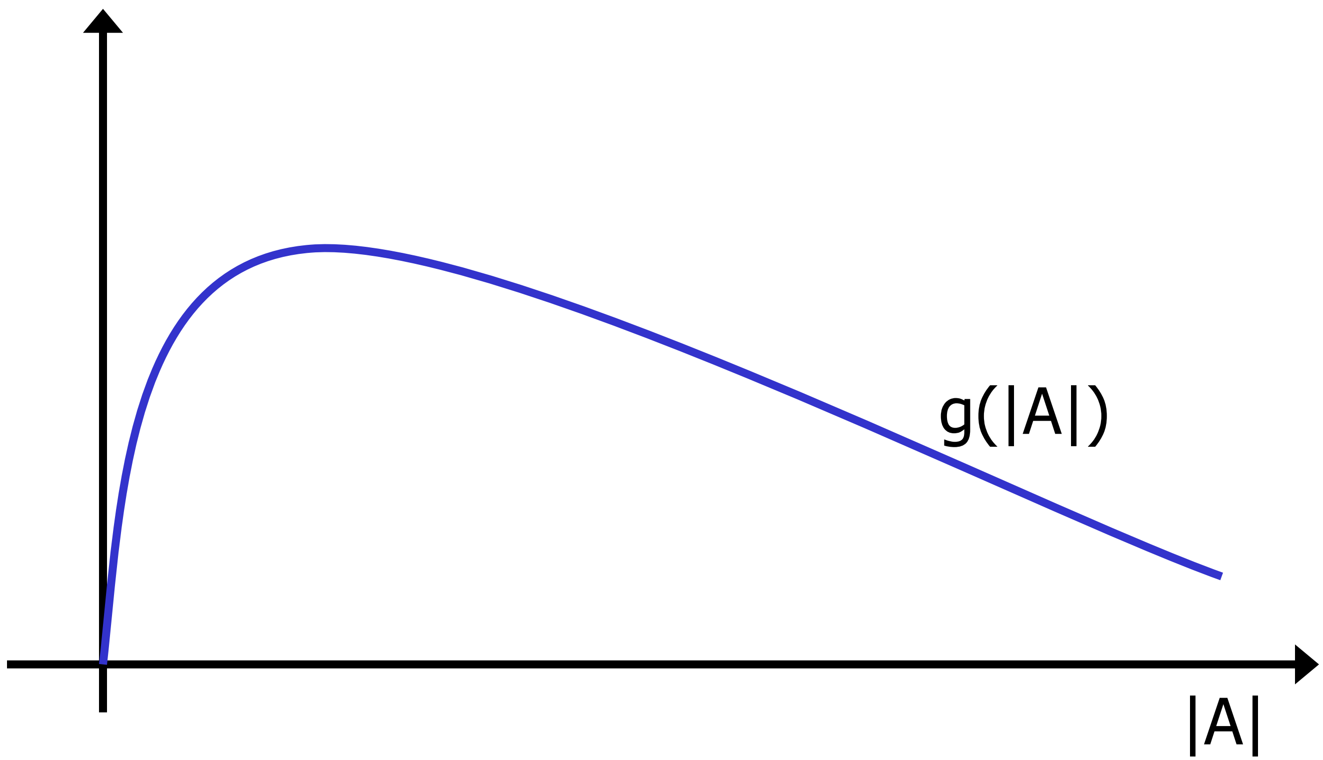
- $F_\theta(A)$ submodular $\Rightarrow \sum_\theta P(\theta) F_\theta(A)$ submodular!
- Multicriterion optimization:
  $F_1,\ldots,F_m$ submodular, $\lambda_i \geq 0 \Rightarrow \sum_i \lambda_i F_i(A)$ submodular

Submodularity and Concavity

Suppose $g: N \rightarrow R$ and $F(A) = g(|A|)$

Then $F(A)$ submodular if and only if g concave!

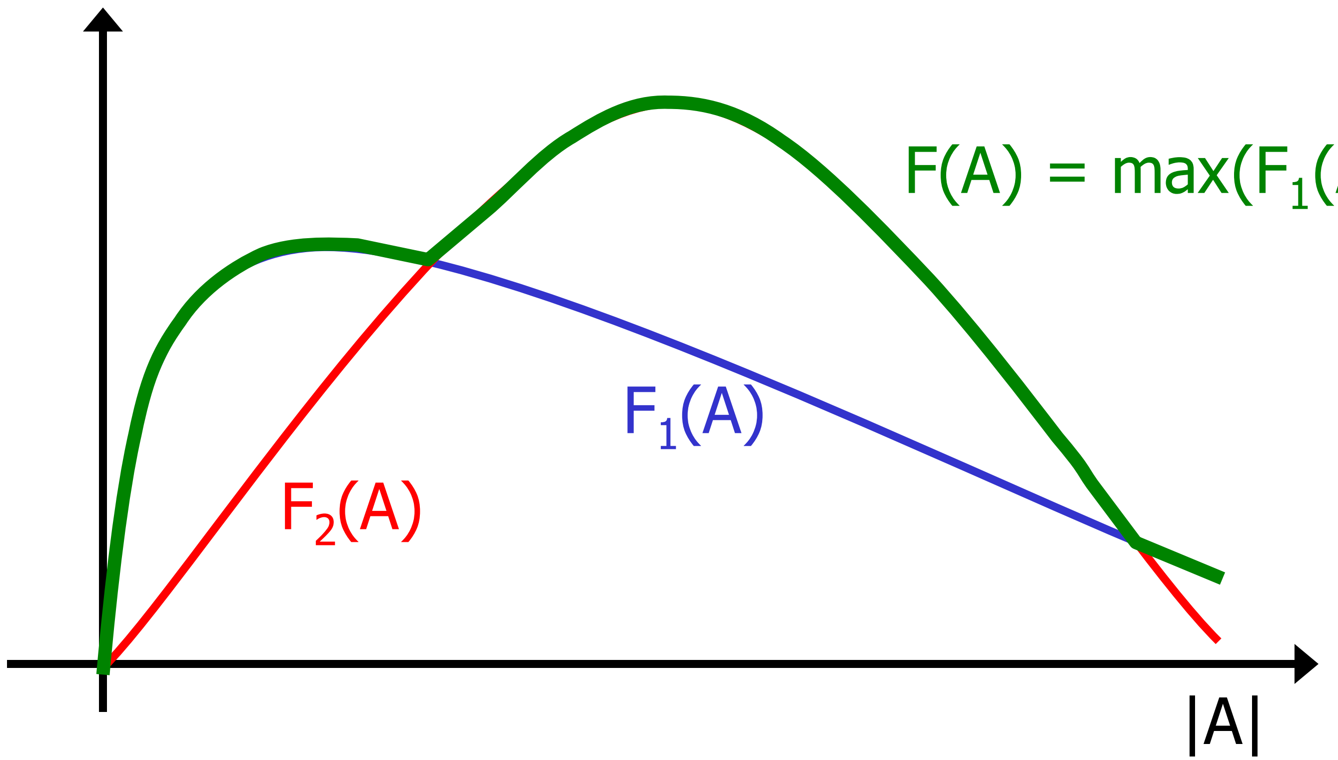
E.g., g could say "buying in bulk is cheaper"



Maximum of submodular functions

Suppose $F_1(A)$ and $F_2(A)$ submodular.

Is $F(A) = \max(F_1(A),F_2(A))$ submodular?



max(F_1,F_2) not submodular in general!

Minimum of submodular functions

Well, maybe $F(A) = \min(F_1(A), F_2(A))$ instead?

	$F_1(A)$	$F_2(A)$
$\emptyset$	0	0
{a}	1	0
{b}	0	1
{a,b}	1	1

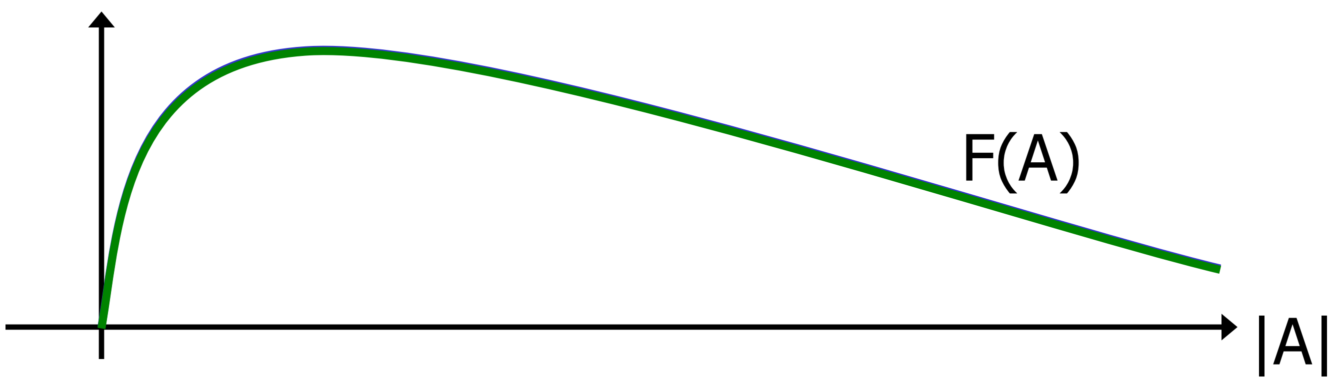
$$F(\{b\}) - F(\emptyset) = 0$$
$$<$$
$$F(\{a,b\}) - F(\{a\}) = 1$$

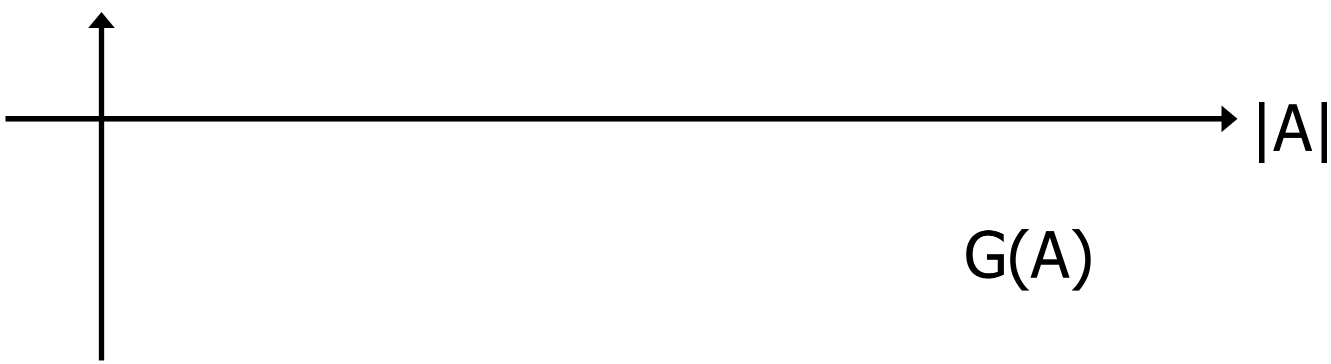
$\min(F_1,F_2)$ not submodular in general!

But stay tuned – we'll address $\min_i F_i$ later!

Duality

- For F submodular on V let $G(A) = F(V) - F(V \setminus A)$
- G is supermodular and called dual to F
- Details about properties in [Fujishige '91]





Tutorial Overview

- Examples and properties of submodular functions ✓
  - Many problems submodular (mutual information, influence, …)
  - SFs closed under positive linear combinations; not under min, max
- Submodularity and convexity
- Minimizing submodular functions
- Maximizing submodular functions
- Extensions and research directions

Submodularity and Convexity

Submodularity and convexity

For $V = \{1,\ldots,n\}$, and $A \subseteq V$, let

$w^A = (w_1^A,\ldots,w_n^A)$ with

$w_i^A = 1$ if $i \in A$, 0 otherwise

Key result [Lovasz '83]: Every submodular function F induces a function g on R^n_+, such that

- $F(A) = g(w^A)$ for all $A \subseteq V$
- $g(w)$ is convex
- $\min_A F(A) = \min_w g(w)$ s.t. $w \in [0,1]^n$

Let's see how one can define g(w)

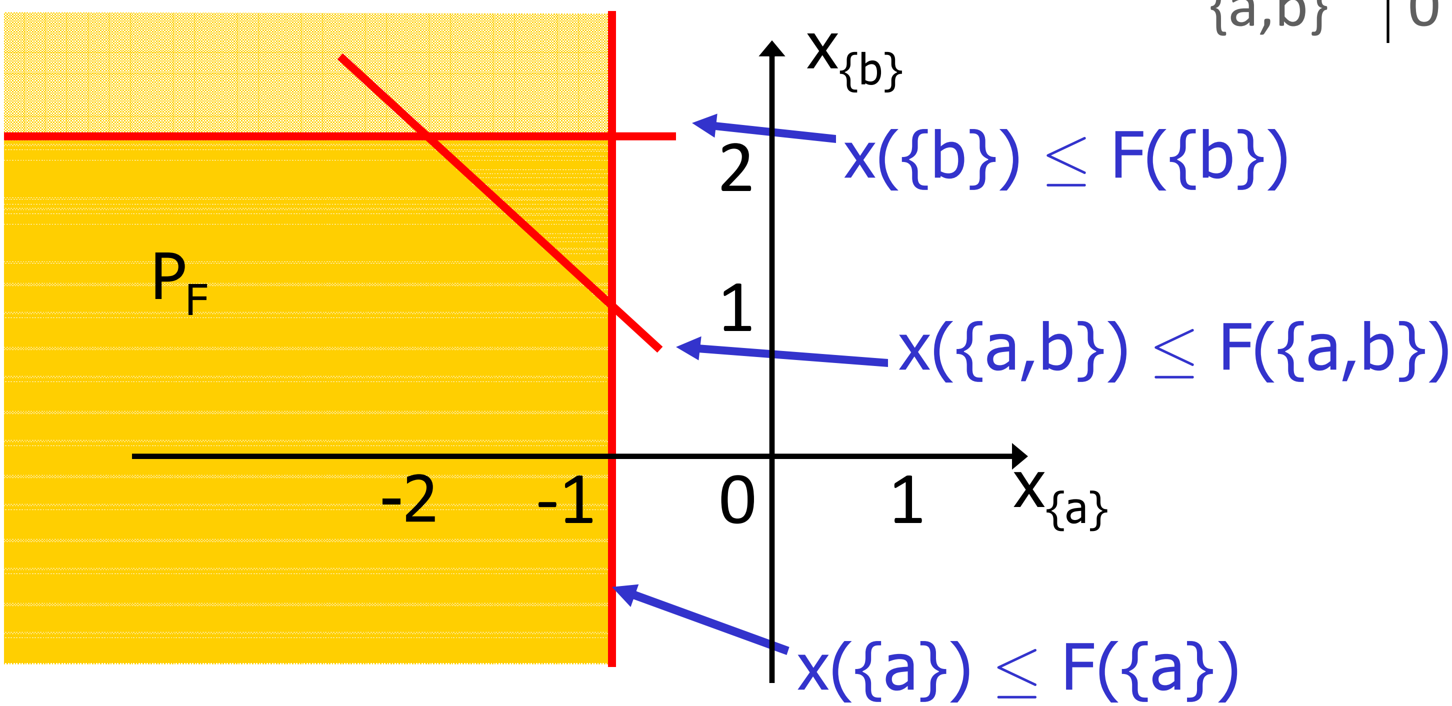
The submodular polyhedron P_F

$$P_F = \{x \in \mathbb{R}^n : x(A) \leq F(A) \text{ for all } A \subseteq V\}$$

$$x(A) = \sum_{i \in A} x_i$$

Example: V = {a,b}

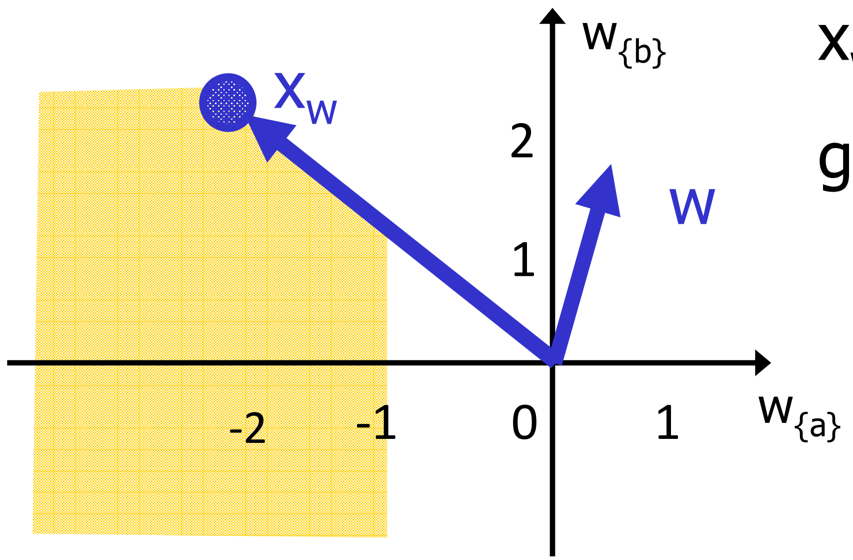
A	F(A)
$\emptyset$	0
{a}	-1
{b}	2
{a,b}	0



Lovasz extension

Claim: $g(w) = \max_{x \in P_F} w^T x$

$P_F = \{x \in \mathbb{R}^n : x(A) \leq F(A) \text{ for all } A \subseteq V\}$

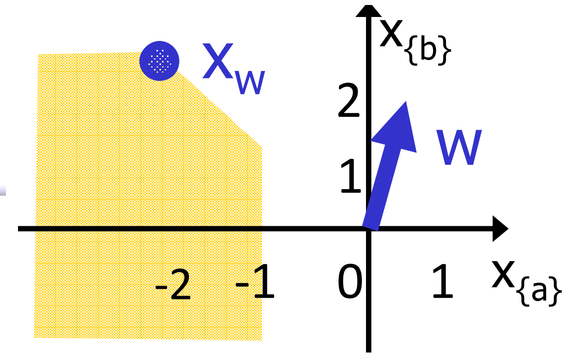


$x_w = \operatorname{argmax}_{x \in P_F} w^T x$

$g(w) = w^T x_w$

Evaluating g(w) requires solving a linear program with exponentially many constraints ☹

Evaluating the Lovasz extension



$g(w) = \max_{x \in P_F} w^T x$

$P_F = \{x \in \mathbb{R}^n : x(A) \leq F(A) \text{ for all } A \subseteq V\}$

Theorem [Edmonds '71, Lovasz '83]:
For any given w, can get optimal solution x_w to the LP using the following greedy algorithm:

1. Order $V=\{e_1,\ldots,e_n\}$ so that $w(e_1) \geq \ldots \geq w(e_n)$
2. Let $x_w(e_i) = F(\{e_1,\ldots,e_i\}) - F(\{e_1,\ldots,e_{i-1}\})$

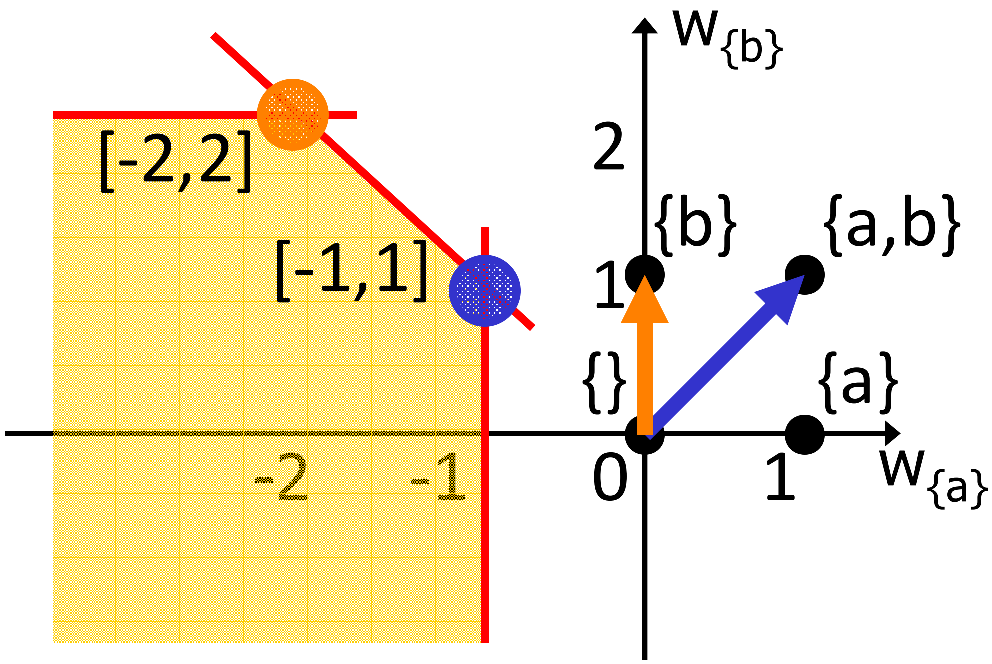
M

Then $w^T x_w = g(w) = \max_{x \in P_F} w^T x$

Sanity check: If $w = w^A$ and $A=\{e_1,\ldots,e_k\}$, then
$w^{A\,T} x^* = \sum_{i=1}^k [F(\{e_1,\ldots,e_i\})-F(\{e_1,\ldots,e_{i-1}\})] = F(A)$

Example: Lovasz extension

$g(w) = \max \{w^T x : x \in P_F\}$



A	F(A)
∅	0
{a}	-1
{b}	2
{a,b}	0

w=[0,1]
want g(w)

Greedy ordering:
$e_1 = b, e_2 = a$
➔ $w(e_1)=1 > w(e_2)=0$

$x_w(e_1)=F(\{b\})-F(\emptyset)=2$

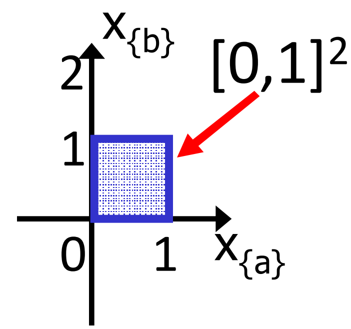
$x_w(e_2)=F(\{b,a\})-F(\{b\})=-2$
➔ $x_w=[-2,2]$

$g([0,1]) = [0,1]^T [-2,2] = 2 = F(\{b\})$

$g([1,1]) = [1,1]^T [-1,1] = 0 = F(\{a,b\})$

Why is this useful?

Theorem [Lovasz '83]:
g(w) attains its minimum in $[0,1]^n$ at a corner!



If we can minimize g on $[0,1]^n$, can minimize F…
(at corners, g and F take same values)

F(A) submodular $\Rightarrow$ g(w) convex (and efficient to evaluate)

$\not\Leftarrow$

Does the converse also hold?

No, consider $g(w_1,w_2,w_3) = \max(w_1,w_2+w_3)$

{a} {b} {c} $F(\{a,b\})-F(\{a\})=0 < F(\{a,b,c\})-F(\{a,c\})=1$

Tutorial Overview

- Examples and properties of submodular functions ✓
  - Many problems submodular (mutual information, influence, …)
  - SFs closed under positive linear combinations; not under min, max
- Submodularity and convexity ✓
  - Every SF induces a convex function with SAME minimum
  - Special properties: Greedy solves LP over exponential polytope
- Minimizing submodular functions
- Maximizing submodular functions
- Extensions and research directions

Minimization of submodular functions

Overview minimization

- Minimizing general submodular functions

- Minimizing symmetric submodular functions

- Applications to Machine Learning

Minimizing a submodular function

Want to solve $\boxed{A^* = \text{argmin}_A F(A)}$

Need to solve

$$\boxed{\begin{array}{l} \min_w \max_x w^T x \\ \text{s.t. } w \in [0,1]^n,\ x \in P_F \end{array}}$$

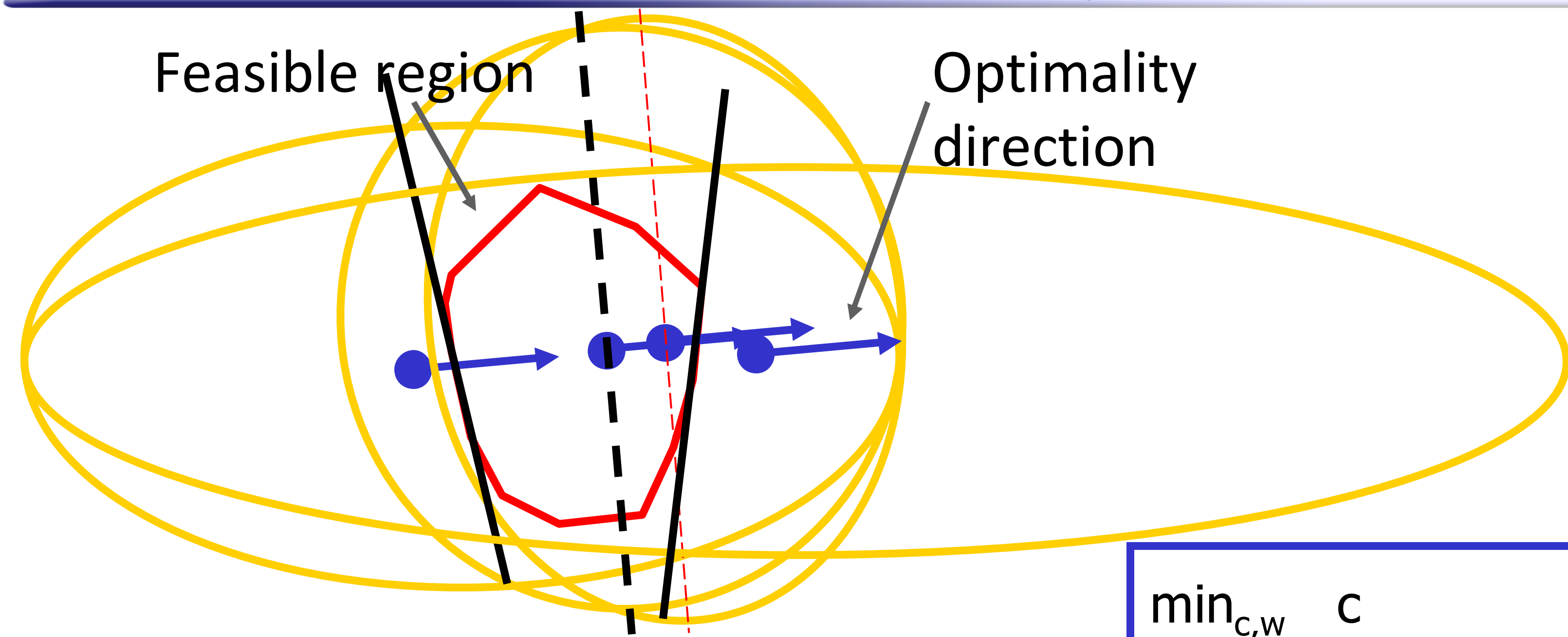
g(w)

Equivalently:

$$\boxed{\begin{array}{lll} \min_{c,w} c & & \\ \text{s.t.} & c \geq w^T x \text{ for all } x \in P_F \\ & w \in [0,1]^n \end{array}}$$

This is an LP with infinitely many constraints!

Ellipsoid algorithm [Grötschel, Lovasz, Schrijver '81]



$$\min_{c,w} \quad c$$
$$\text{s.t. } c \geq w^T x \text{ for all } x \in P_F$$
$$w \in [0,1]^n$$

Separation oracle: Find most violated constraint:

$$\max_x w^T x - c \quad \text{s.t. } x \in P_F$$

Can solve separation using the greedy algorithm!!

➔ Ellipsoid algorithm minimizes SFs in poly-time!

Minimizing submodular functions

Ellipsoid algorithm not very practical

Want combinatorial algorithm for minimization!

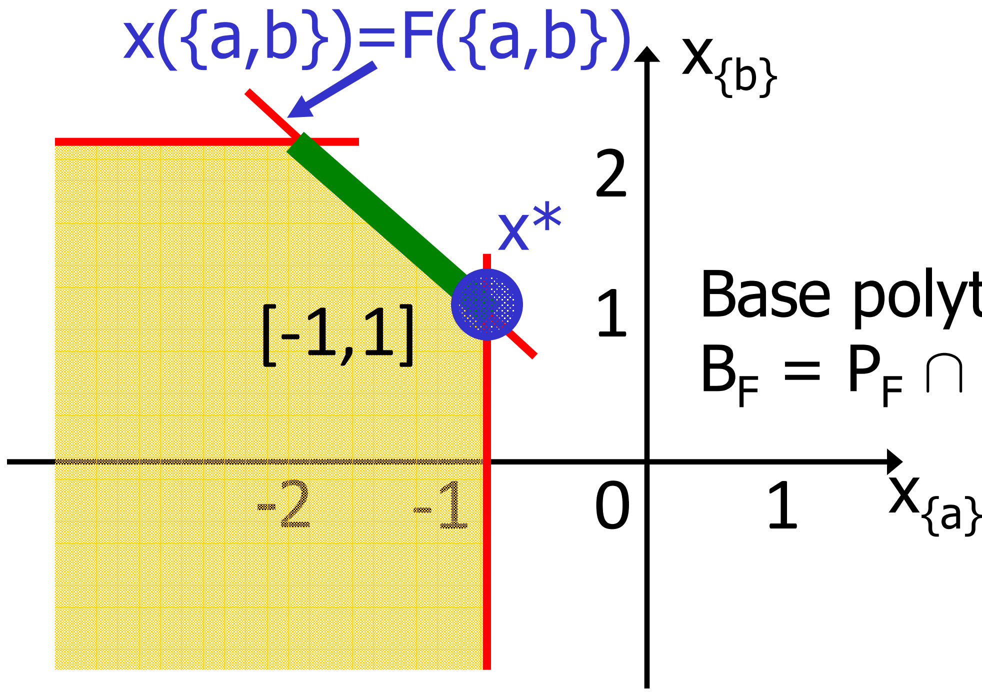
Theorem [Iwata (2001)]

There is a fully combinatorial, strongly polynomial algorithm for minimizing SFs, that runs in time

$$O(n^8 \log^2 n)$$

Polynomial-time = Practical ???

A more practical alternative? [Fujishige '91, Fujishige et al '06]



Base polytope:
$B_F = P_F \cap \{x(V) = F(V)\}$

A	F(A)
∅	0
{a}	-1
{b}	2
{a,b}	0

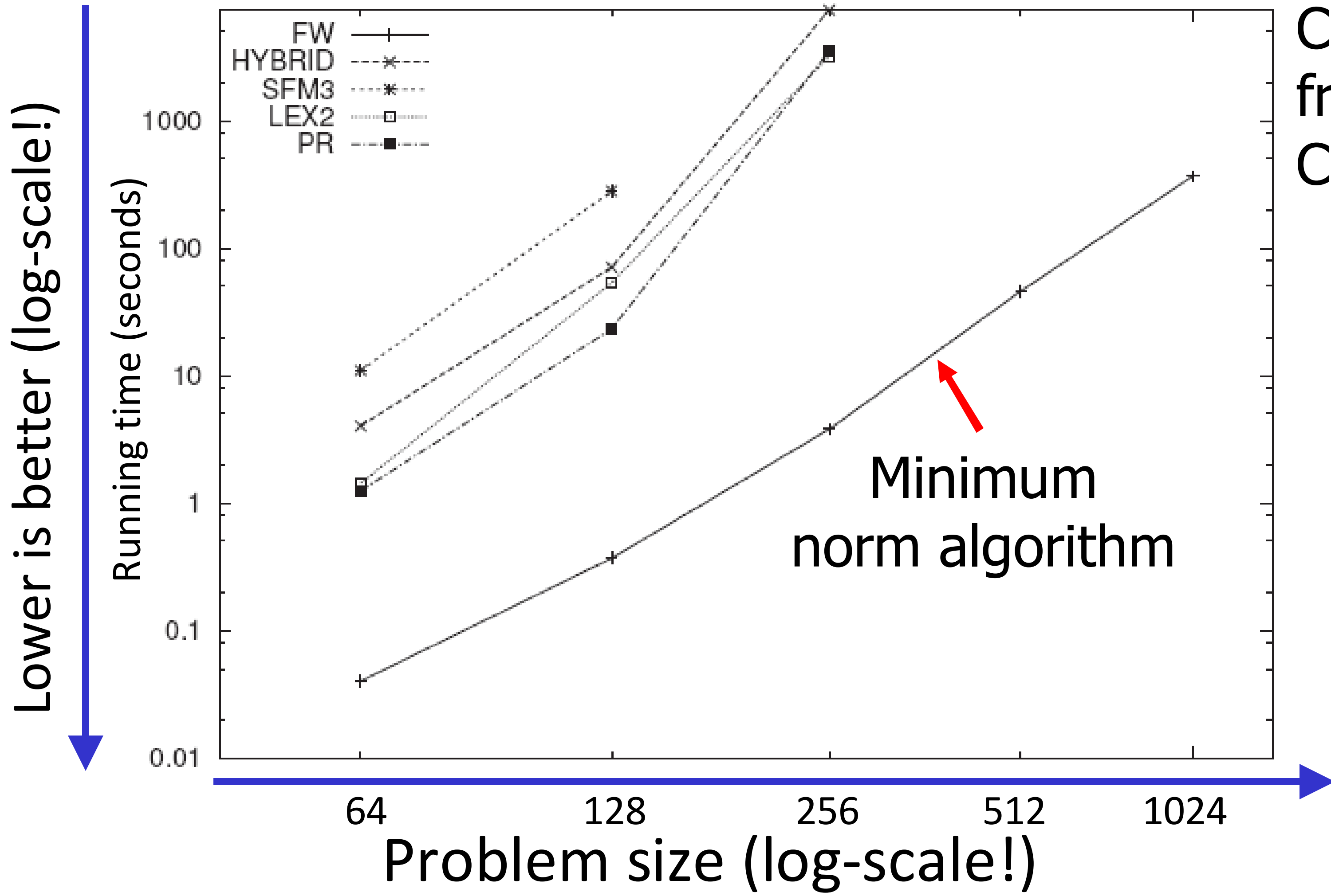
<u>Minimum norm algorithm:</u> M

1. Find $x^* = \text{argmin } ||x||_2$ s.t. $x \in B_F$
2. Return $A^* = \{i: x^*(i) < 0\}$

$x^*=[-1,1]$
$A^*=\{a\}$

Theorem [Fujishige '91]: A* is an optimal solution!
Note: Can solve 1. using Wolfe's algorithm
Runtime finite but unknown!! ☹

Empirical comparison
[Fujishige et al '06]

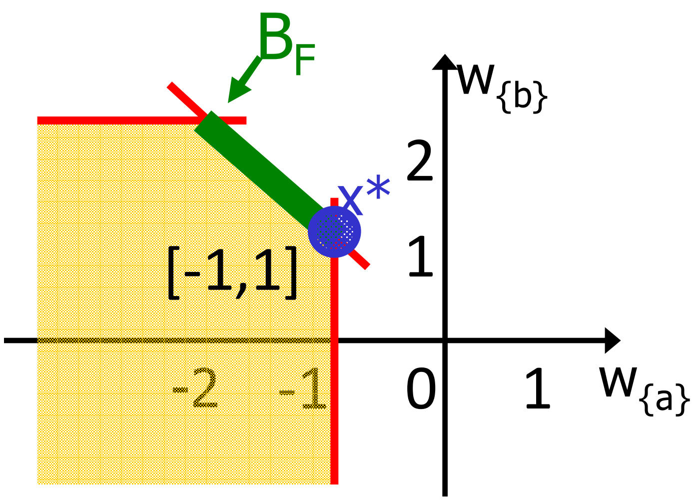


Cut functions from DIMACS Challenge

Minimum norm algorithm orders of magnitude faster!

Our implementation can solve n = 10k in < 6 minutes!

Checking optimality (duality)



A	F(A)
$\emptyset$	0
{a}	-1
{b}	2
{a,b}	0

Base polytope:

$B_F = P_F \cap \{x(V) = F(V)\}$

Theorem [Edmonds '70]

$\min_A F(A) = \max_x \{x^-(V) : x \in B_F\}$

where $x^-(s) = \min \{x(s), 0\}$

$A = \{a\}, F(A) = -1$

$w = [1,0]$

$x_w = [-1,1]$

$x_w^- = [-1,0]$

$x_w^-(V) = -1$

$\Rightarrow$ A optimal!

Testing how close A' is to $\min_A F(A)$

1. Run greedy algorithm for $w=w_{A'}$ to get x_w
2. $F(A') \geq \min_A F(A) \geq x_w^-(V)$

Overview minimization

- Minimizing general submodular functions ✓
  - Can minimizing in polytime using ellipsoid method
  - Combinatorial, strongly polynomial algorithm O(n^8)
  - Practical alternative: Minimum norm algorithm?
- Minimizing symmetric submodular functions
- Applications to Machine Learning

What if we have special structure?

Worst-case complexity of best known algorithm: $O(n^8 \log^2 n)$

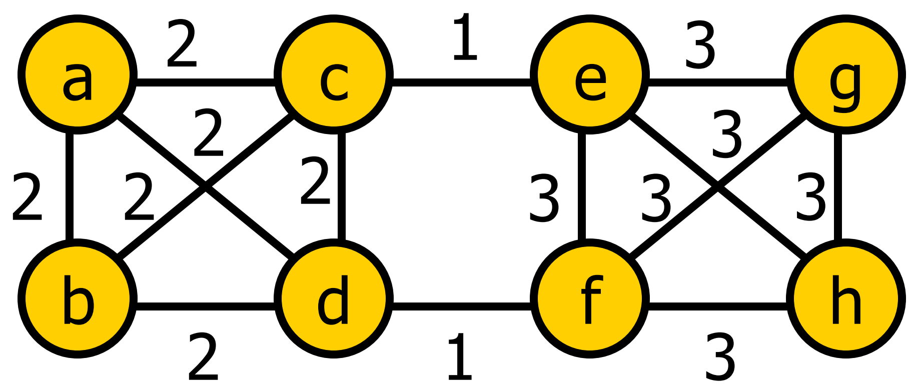
Can we do better for special cases?

Example (again): Given RVs $X_1,\dots,X_n$

$$F(A) = I(X_A; X_{V \setminus A}) = I(X_{V \setminus A}; X_A) = F(V \setminus A)$$

Functions F with $F(A) = F(V \setminus A)$ for all A are symmetric

Another example: Cut functions



$V=\{a,b,c,d,e,f,g,h\}$

$$F(A) = \sum \{w_{s,t} : s \in A,\ t \in V \setminus A\}$$

Example: $F(\{a\})=6$; $F(\{c,d\})=10$; $F(\{a,b,c,d\})=2$

Cut function is symmetric and submodular!

Minimizing symmetric functions

- For any A, submodularity implies

$$
\begin{aligned}
2\,F(A) &= F(A) + F(V \setminus A) \\
&\geq F(A \cap (V \setminus A)) + F(A \cup (V \setminus A)) \\
&= F(\emptyset) + F(V) \\
&= 2\,F(\emptyset) = 0
\end{aligned}
$$

- Hence, any symmetric SF attains minimum at $\emptyset$
- In practice, want nontrivial partition of V into A and $V \setminus A$, i.e., require that A is neither $\emptyset$ of V

$$\boxed{\text{Want } A^* = \operatorname{argmin} F(A) \quad \text{s.t.} \quad 0 < |A| < n}$$

There is an efficient algorithm for doing that! ☺

Queyranne's algorithm (overview) [Queyranne '98]

M

Theorem: There is a fully combinatorial, strongly polynomial algorithm for solving

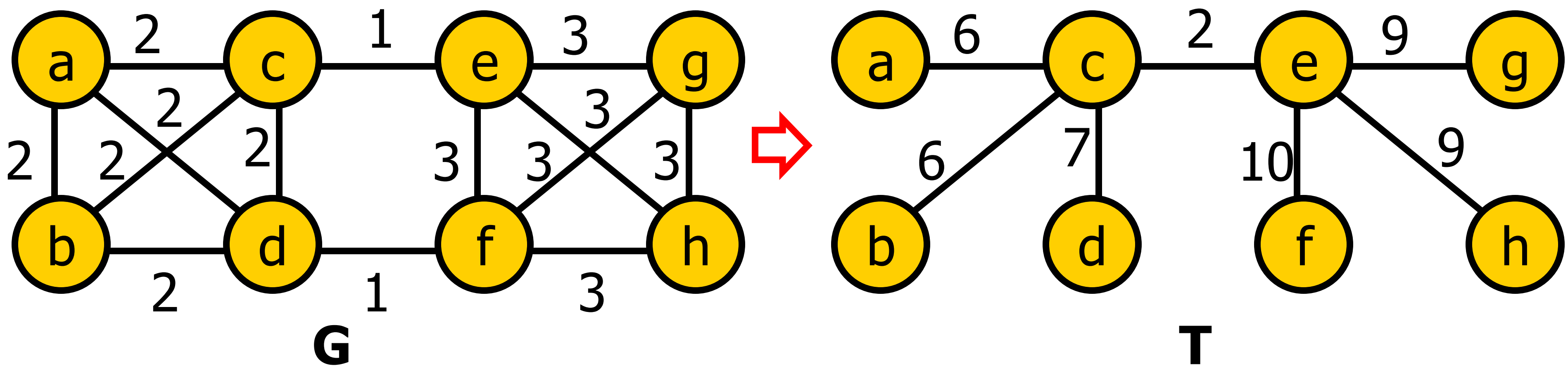
$$A^* = \text{argmin}_A F(A) \quad \text{s.t.} \quad 0<|A|<n$$

for symmetric submodular functions A

- Runs in time $O(n^3)$ [instead of $O(n^8)$...]

Note: also works for "posimodular" functions:
F posimodular ⇔ $A,B \subseteq V$: $F(A)+F(B) \geq F(A \setminus B)+F(B \setminus A)$

Gomory Hu trees



A tree T is called Gomory-Hu (GH) tree for SF F
if for any s, t $\in$ V it holds that

$$\min \{F(A): s \in A \text{ and } t \notin A\} = \min \{w_{i,j}: (i,j) \text{ is an edge on the s-t path in } T\}$$

"min s-t-cut in T = min s-t-cut in G"

> **Theorem** [Queyranne '93]:
> GH-trees exist for any symmetric SF F!

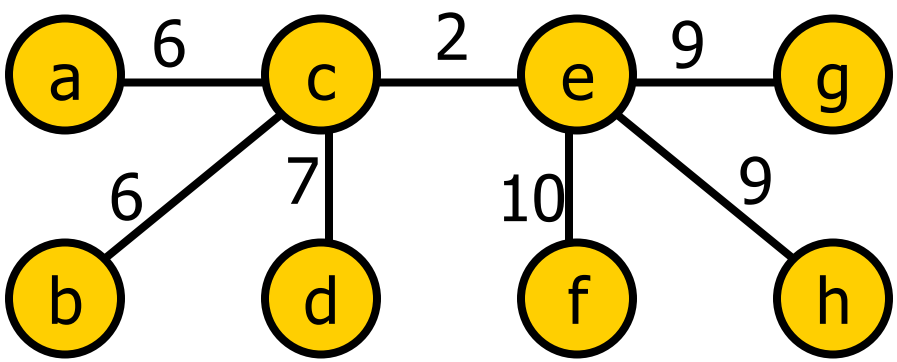
Expensive to find one in general! ☹

Pendent pairs

For function F on V, s,t$\in$ V: (s,t) is pendent pair if

$$\{s\} \in \operatorname{argmin}_A F(A) \quad \text{s.t.} \quad s\in A,\ t\notin A$$

Pendent pairs always exist:



Gomory-Hu tree **T**

Take any leaf s and neighbor t, then (s,t) is pendent!

E.g., (a,c), (b,c), (f,e), …

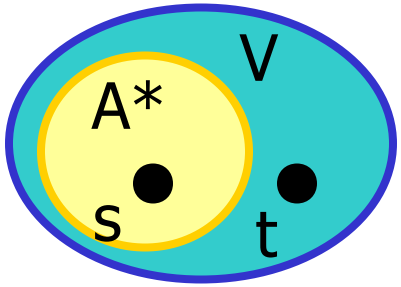
Theorem [Queyranne '95]: Can find pendent pairs in $O(n^2)$ (without needing GH-tree!)

Why are pendent pairs useful?

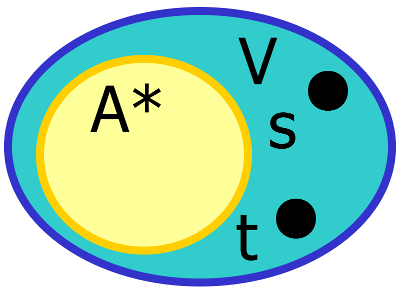
- Key idea: Let (s,t) pendent, A* = argmin F(A)

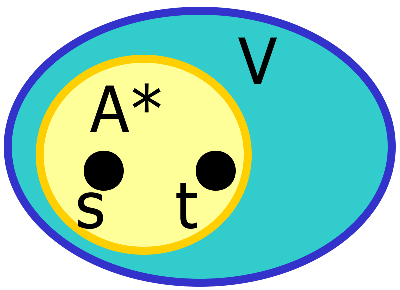
  Then EITHER

  - s and t separated by A*, e.g., $s \in A^*$, $t \notin A^*$.
    But then A*={s}!! OR



  - s and t are not separated by A*



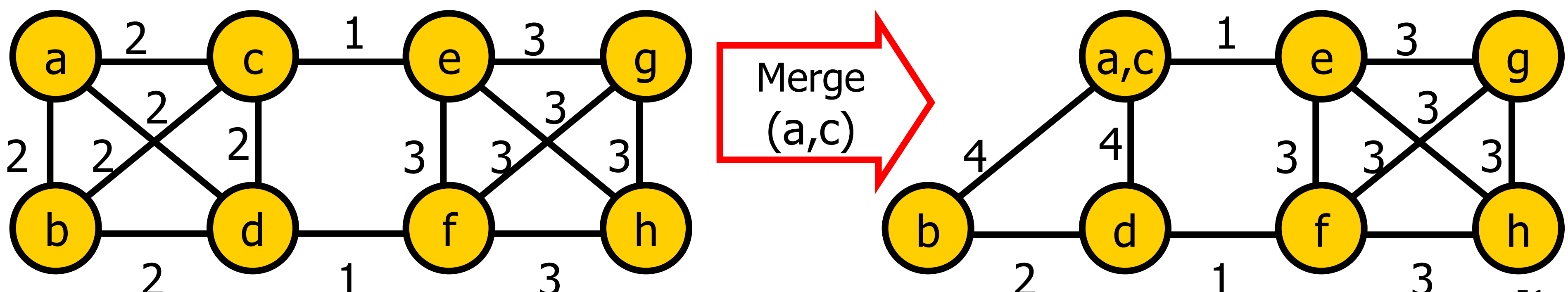


Then we can merge s and t…

Merging

- Suppose F is a symmetric SF on V,
  and we want to merge pendent pair (s,t)
- Key idea: “If we pick s, get t for free”
  - $V' = V \setminus \{t\}$
  - $F'(A) \quad = F(A \cup \{t\}) \qquad \text{if } s \in A, \text{ or}$
    $\qquad\qquad = F(A) \quad \text{if } s \notin A$

Lemma: F’ is still symmetric and submodular!



Queyranne's algorithm

Input: symmetric SF F on V, |V|=n

Output: $A^* = \operatorname{argmin} F(A)$ s.t. $0 < |A| < n$

Initialize $F' \leftarrow F$, and $V' \leftarrow V$

For i = 1:n-1

- $(s,t) \leftarrow \text{pendentPair}(F',V')$
- $A_i = \{s\}$
- $(F',V') \leftarrow \text{merge}(F',V',s,t)$

Return $\operatorname{argmin}_i F(A_i)$

Running time: $O(n^3)$ function evaluations

Note: Finding pendent pairs

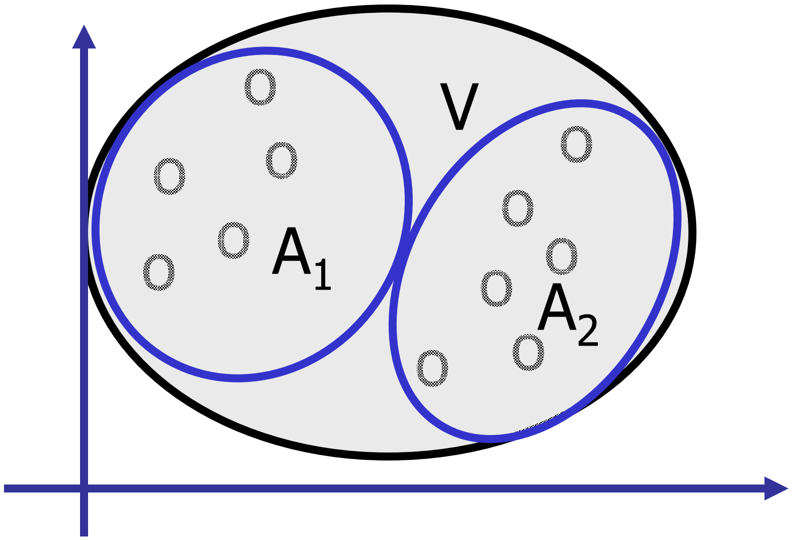
1. Initialize $v_1 \leftarrow x$ (x is arbitrary element of V)
2. For i = 1 to n-1 do
   1. $W_i \leftarrow \{v_1,\ldots,v_i\}$
   2. $v_{i+1} \leftarrow \text{argmin}_v \, F(W_i \cup \{v\}) - F(\{v\})$ s.t. $v \in V \setminus W_i$
3. Return pendent pair (v_{n-1}, v_n)

Requires $O(n^2)$ evaluations of F

Overview minimization

- Minimizing general submodular functions ✓
  - Can minimizing in polytime using ellipsoid method
  - Combinatorial, strongly polynomial algorithm $O(n^8)$
  - Practical alternative: Minimum norm algorithm?
- Minimizing symmetric submodular functions ✓
  - Many useful submodular functions are symmetric
  - Queyranne's algorithm minimize symmetric SFs in $O(n^3)$
- Applications to Machine Learning

Application: Clustering
[Narasimhan, Jojic, Bilmes NIPS '05]



Group data points V into "homogeneous clusters"

Find a partition $V = A_1 \cup \ldots \cup A_k$ that minimizes

$$F(A_1, \ldots, A_k) = \sum_i E(A_i)$$

"Inhomogeneity of A_i"
Examples for E(A):

- Entropy H(A)
- Cut function

Special case: k = 2. Then $F(A) = E(A) + E(V \setminus A)$ is symmetric!

If E is submodular, can use Queyranne's algorithm! ☺

What if we want k>2 clusters?
[Zhao et al '05, Narasimhan et al '05]

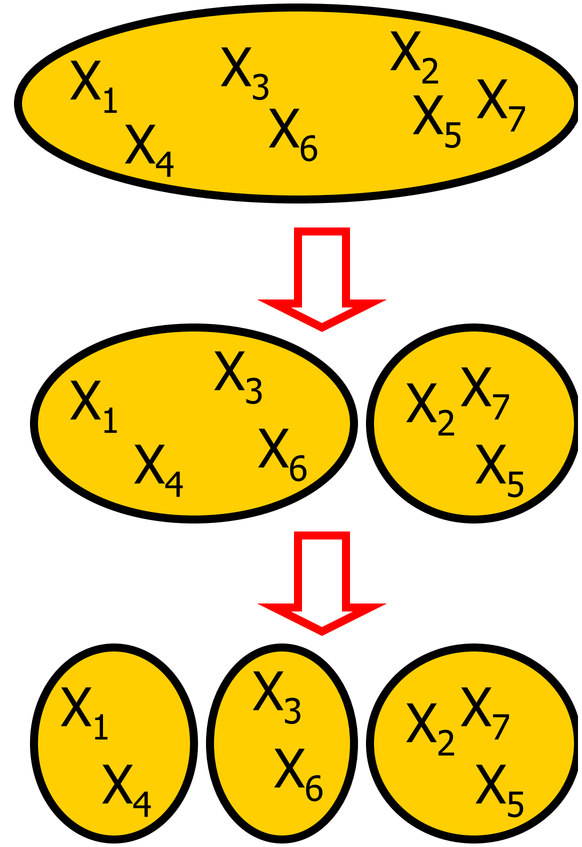
M

<u>Greedy Splitting algorithm</u>

Start with partition $P = \{V\}$

For $i = 1$ to $k-1$

- For each member $C_j \in P$ do
  - split cluster C_j:
    $A^* = \text{argmin}\ E(A) + E(C_j \setminus A)$ s.t. $0<|A|<|C_j|$
  - $P_j \leftarrow P \setminus \{C_j\} \cup \{A, C_j \setminus A\}$
    Partition we get by splitting j-th cluster
- $P \leftarrow \text{argmin}_j\ F(P_j)$



Theorem: $F(P) \leq (2\text{-}2/k)\ F(P_{opt})$

Example: Clustering species [Narasimhan et al '05]

Species X	ATGCCTGA
Species Y	TGCCTAGTGGA
Species Z	TGGAGCCTTGA

Common genetic information = #of common substrings:

$$I_{\mathsf{CG}}(X;Y) = |\{\mathsf{TGC}, \mathsf{GCC}, \mathsf{CCT}, \mathsf{GCCT}, \mathsf{TGCC}, \mathsf{TGCCT}\}| = 6$$

$$I_{\mathsf{CG}}(X;Z) = |\{\mathsf{GCC}, \mathsf{CCT}, \mathsf{GCCT}\}| = 3$$

Can easily extend to sets of species

$$I_{\mathsf{CG}}(X;\{Y,Z\}) = |\{\mathsf{TGC}, \mathsf{GCC}, \mathsf{CCT}, \mathsf{TGCC}, \mathsf{GCCT}, \mathsf{TGCCT}\}| = 6$$

Example: Clustering species [Narasimhan et al '05]

- The common genetic information I_{CG}
  - does not require alignment
  - captures genetic similarity
  - is smallest for maximally evolutionarily diverged species
  - is a symmetric submodular function! ☺

- Greedy splitting algorithm yields phylogenetic tree!

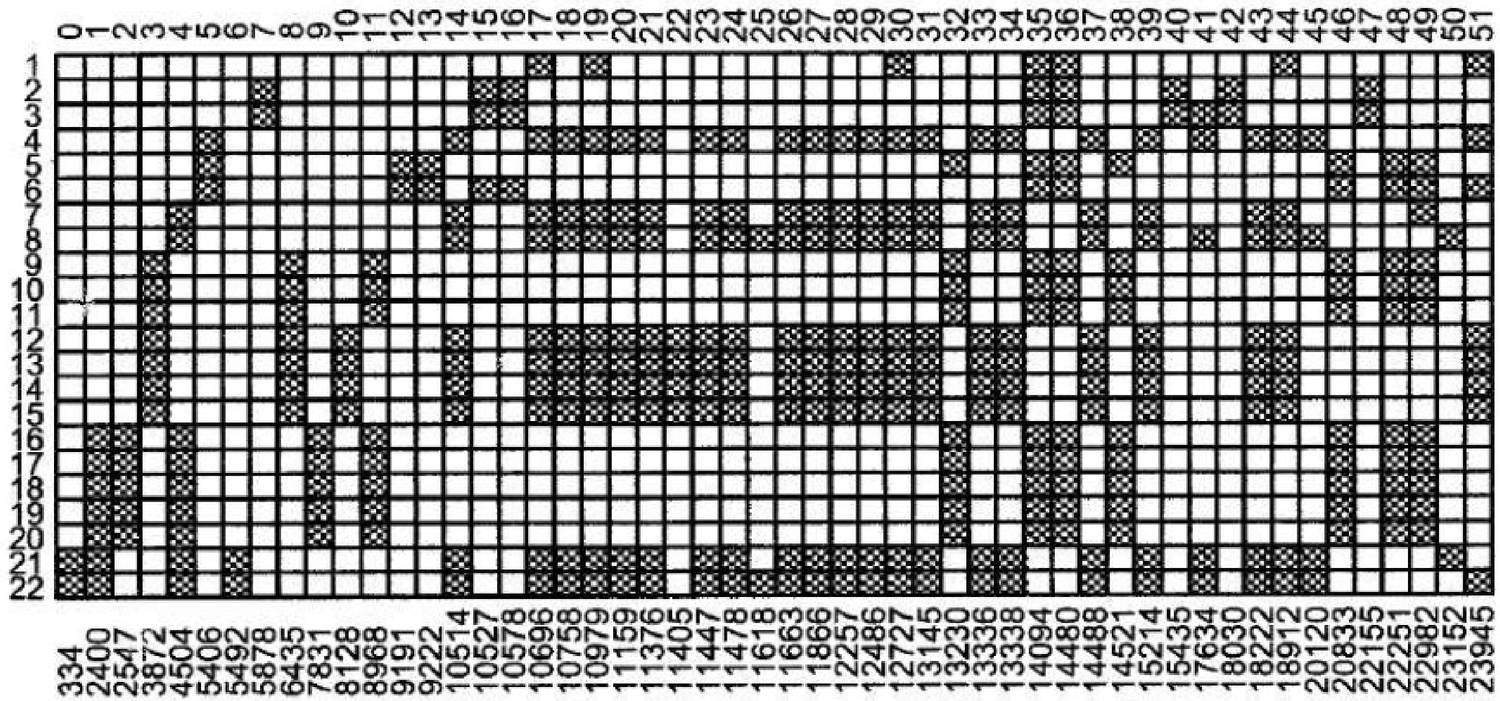
sense learn act

Example: SNPs [Narasimhan et al '05]

- Study human genetic variation (for personalized medicine, …)
- Most human variation due to point mutations that occur once in human history at that base location:

  Single Nucleotide Polymorphisms (SNPs)
- Cataloging all variation too expensive ($10K-$100K per individual!!)

SNPs in the ACE gene
[Narasimhan et al ‘05]

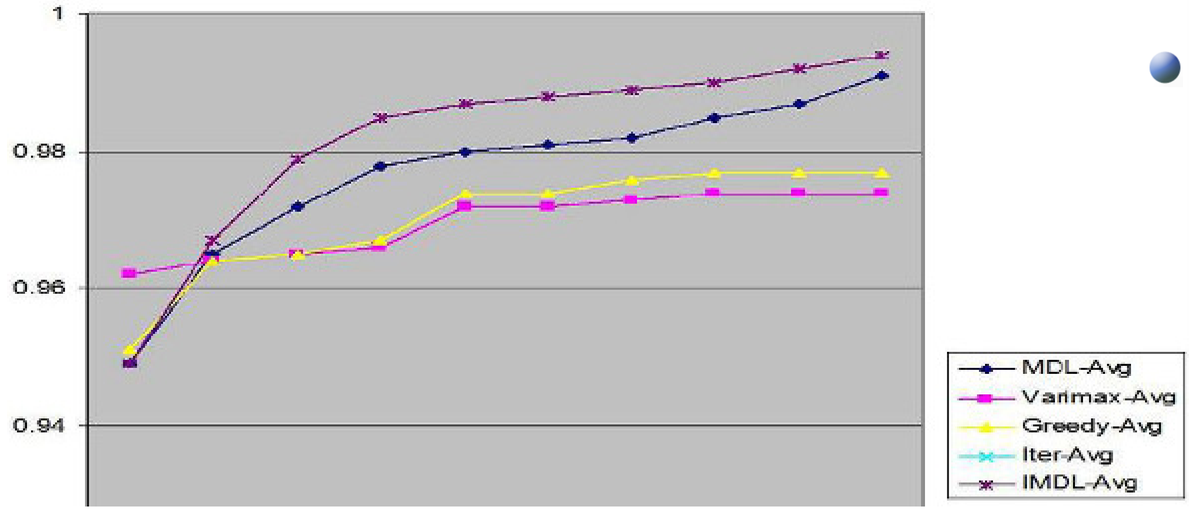


Rows: Individuals. Columns: SNPs.

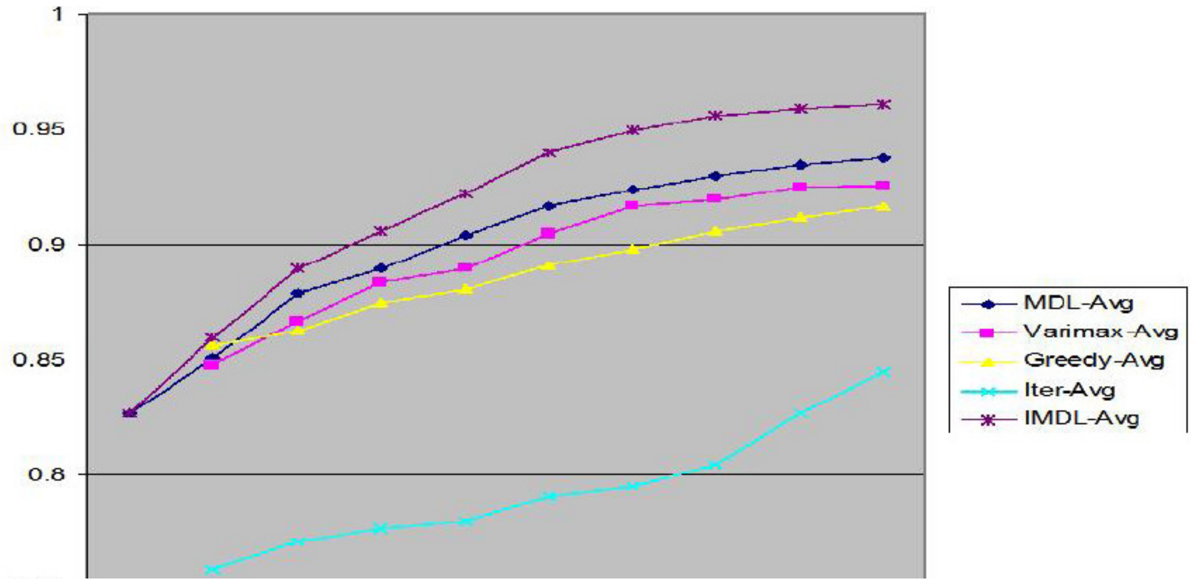
Which columns should we pick to reconstruct the rest?

Can find near-optimal clustering (Queyranne’s algorithm)

Reconstruction accuracy [Narasimhan et al '05]

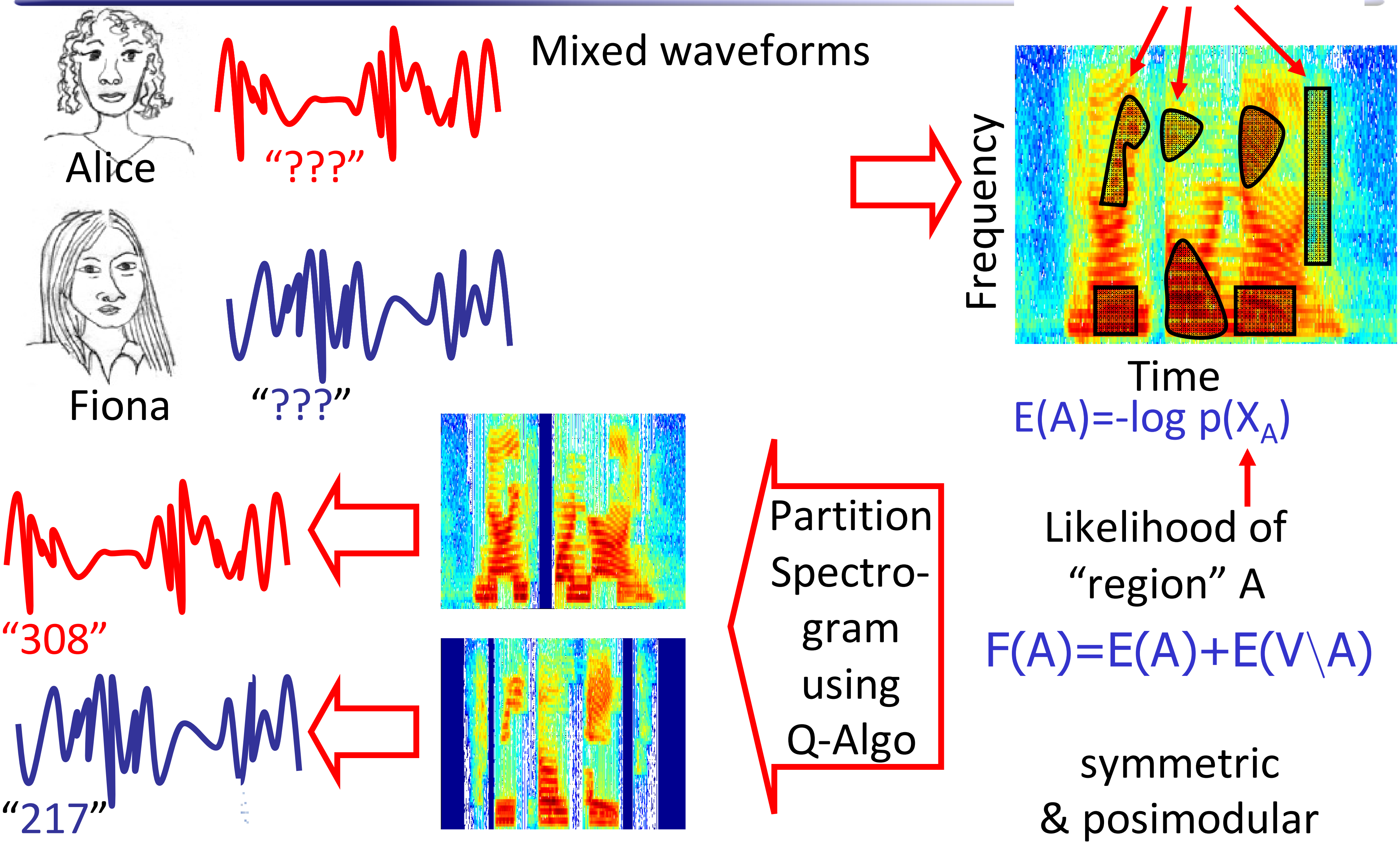


# of clusters

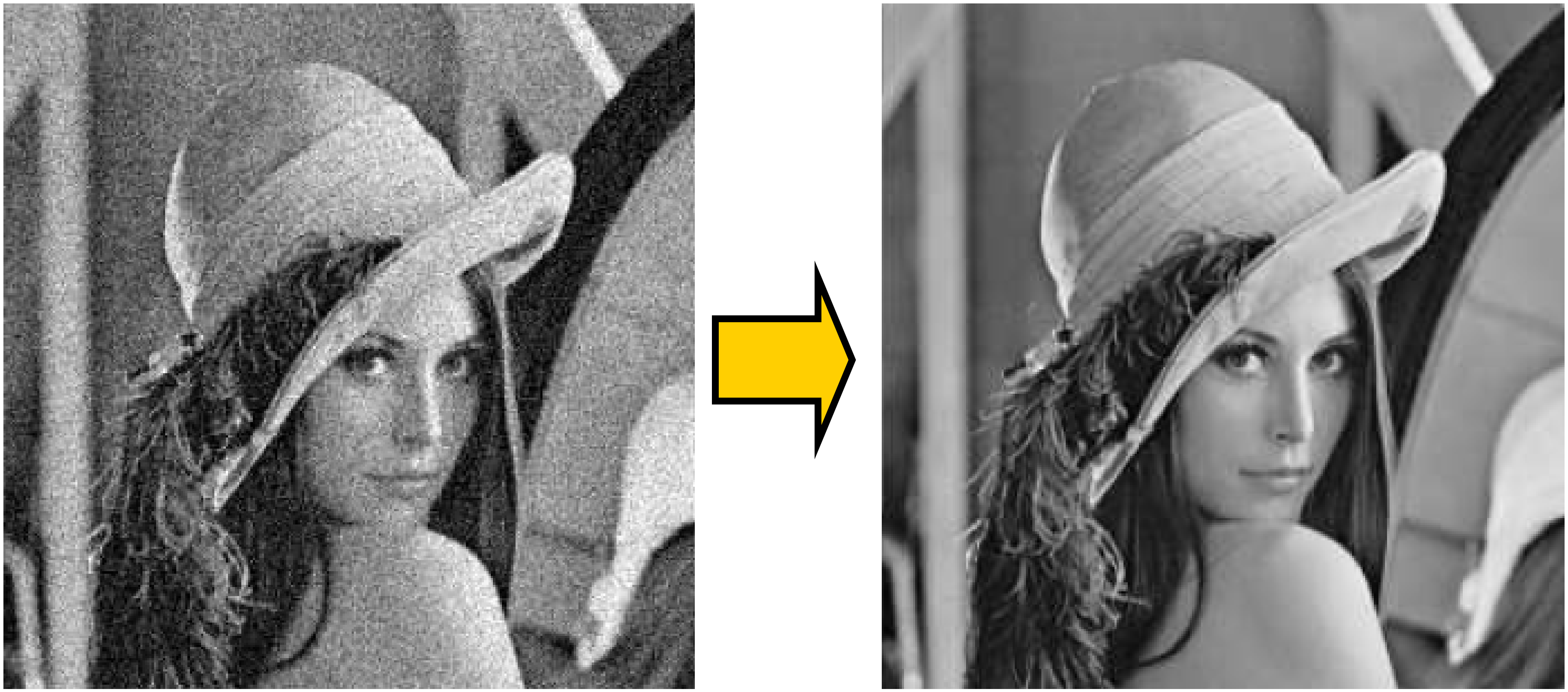


- Comparison with clustering based on
  - Entropy
  - Prediction accuracy
  - Pairwise correlation
  - PCA

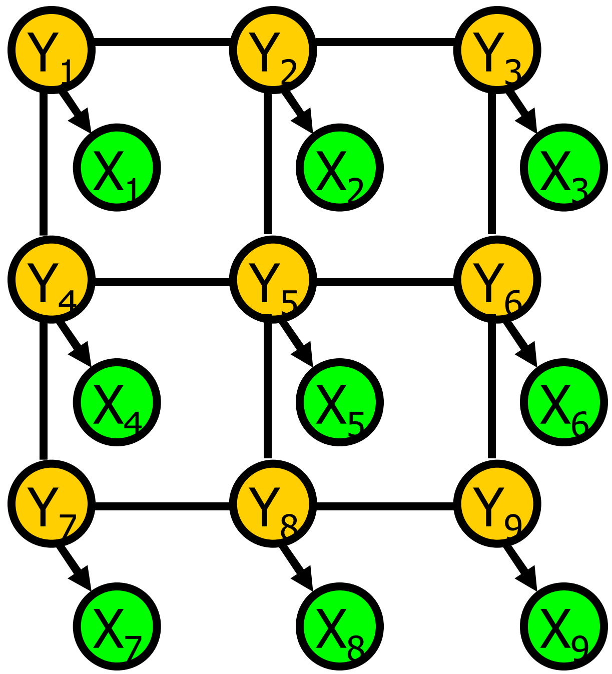
Example: Speaker segmentation [Reyes-Gomez, Jojic '07]



Example: Image denoising



Example: Image denoising



X_i: noisy pixels

Y_i: "true" pixels

Pairwise Markov Random Field

$$P(x_1,\ldots,x_n,y_1,\ldots,y_n) = \prod_{i,j} \psi_{i,j}(y_i,y_j) \prod_i \phi_i(x_i,y_i)$$

Want $\arg\max_y P(y \mid x)$

$= \arg\max_y \log P(x,y)$

$= \arg\min_y \sum_{i,j} E_{i,j}(y_i,y_j) + \sum_i E_i(y_i)$

$$E_{i,j}(y_i,y_j) = -\log \psi_{i,j}(y_i,y_j)$$

When is this MAP inference efficiently solvable (in high treewidth graphical models)?

MAP inference in Markov Random Fields

[Kolmogorov et al, PAMI '04, see also: Hammer, Ops Res '65]

Energy $E(y) = \sum_{i,j} E_{i,j}(y_i, y_j) + \sum_i E_i(y_i)$

Suppose y_i are binary, define

$F(A) = E(y^A)$ where $y^A_i = 1$ iff $i \in A$

Then $\min_y E(y) = \min_A F(A)$

Theorem

MAP inference problem solvable by graph cuts

$\Leftrightarrow$ For all i,j: $E_{i,j}(0,0)+E_{i,j}(1,1) \leq E_{i,j}(0,1)+E_{i,j}(1,0)$

$\Leftrightarrow$ each $E_{i,j}$ is submodular

"Efficient if prefer that neighboring pixels have same color"

Constrained minimization

Have seen: if F submodular on V, can solve

$$\boxed{A^* = \operatorname{argmin} F(A) \qquad \text{s.t.} \qquad A \in V}$$

What about

$$\boxed{A^* = \operatorname{argmin} F(A) \qquad \text{s.t.} \qquad A \in V \text{ and } |A| \leq k}$$

E.g., clustering with minimum # points per cluster, …

In general, not much known about constrained minimization ☹

However, can do

- A^*=argmin F(A) s.t. $0<|A|< n$
- A^*=argmin F(A) s.t. $|A|$ is odd/even [Goemans&Ramakrishnan '95]
- A^*=argmin F(A) s.t. $A \in$ argmin G(A) for G submodular [Fujishige '91]

Overview minimization

- Minimizing general submodular functions ✓
  - Can minimizing in polytime using ellipsoid method
  - Combinatorial, strongly polynomial algorithm $O(n^8)$
  - Practical alternative: Minimum norm algorithm?
- Minimizing symmetric submodular functions ✓
  - Many useful submodular functions are symmetric
  - Queyranne's algorithm minimize symmetric SFs in $O(n^3)$
- Applications to Machine Learning ✓
  - Clustering [Narasimhan et al' 05]
  - Speaker segmentation [Reyes-Gomez & Jojic '07]
  - MAP inference [Kolmogorov et al '04]

Tutorial Overview

- Examples and properties of submodular functions ✓
  - Many problems submodular (mutual information, influence, …)
  - SFs closed under positive linear combinations; not under min, max
- Submodularity and convexity ✓
  - Every SF induces a convex function with SAME minimum
  - Special properties: Greedy solves LP over exponential polytope
- Minimizing submodular functions ✓
  - Minimization possible in polynomial time (but $O(n^8)$…)
  - Queyranne's algorithm minimizes symmetric SFs in $O(n^3)$
  - Useful for clustering, MAP inference, structure learning, …
- Maximizing submodular functions
- Extensions and research directions

Maximizing submodular functions

Maximizing submodular functions

Minimizing convex functions:
Polynomial time solvable!

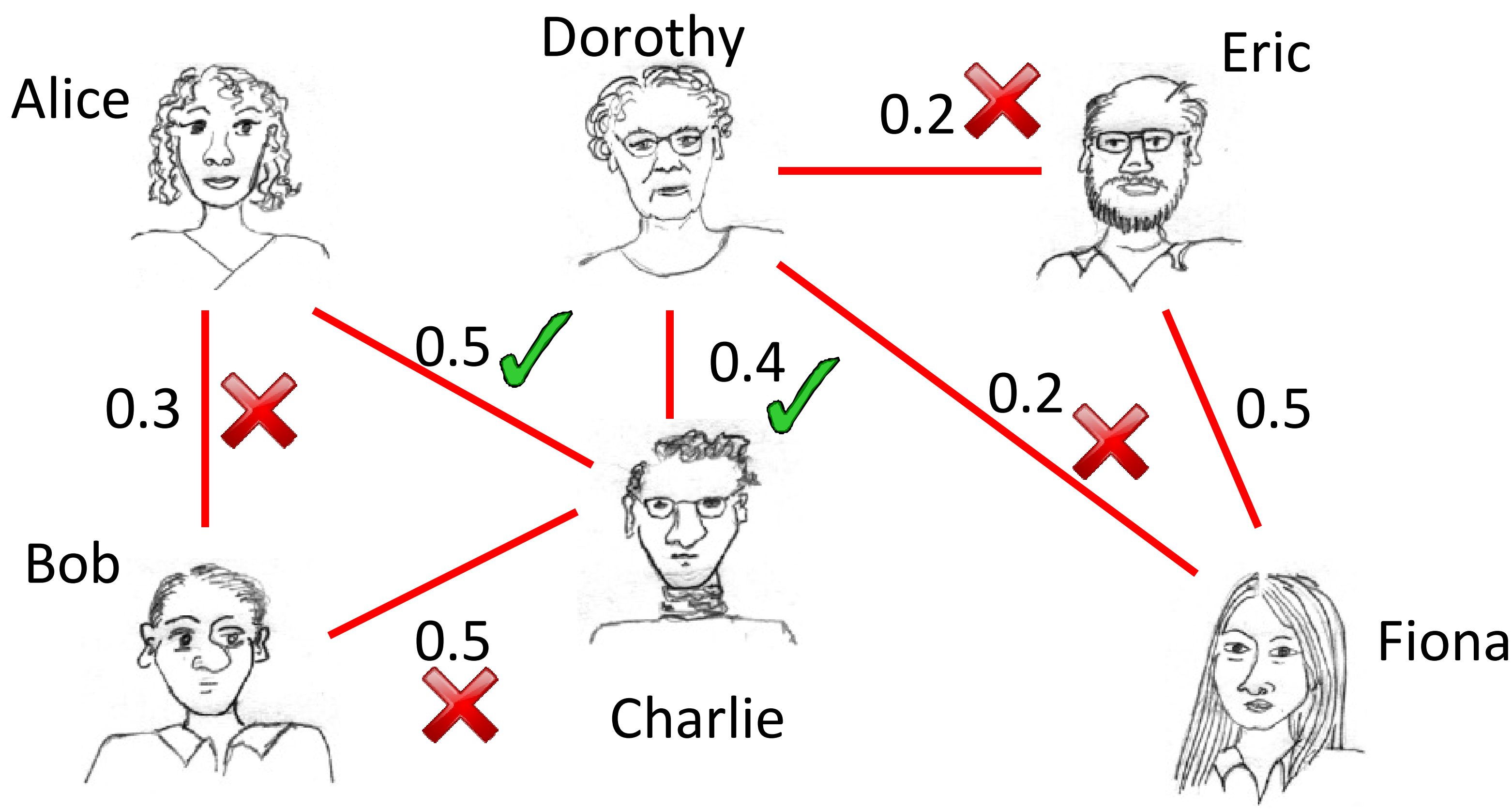
Minimizing submodular functions:
Polynomial time solvable!

Maximizing convex functions:
NP hard!

Maximizing submodular functions:
NP hard!

But can get approximation guarantees ☺

Maximizing influence
[Kempe, Kleinberg, Tardos KDD '03]



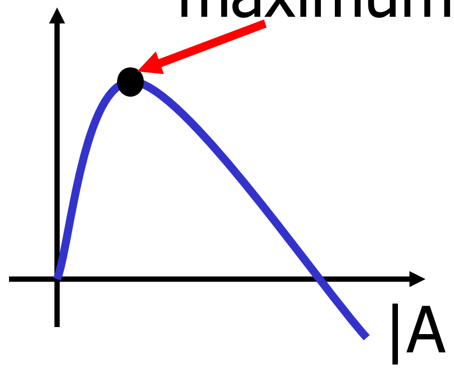
- $F(A)$ = Expected #people influenced when targeting A
- F monotonic: If $A \subseteq B$: $F(A) \leq F(B)$
  Hence $V = \text{argmax}_A F(A)$

More interesting: $\text{argmax}_A F(A) - \text{Cost}(A)$

Maximizing non-monotonic functions

- Suppose we want for not monotonic F

$$\boxed{A^* = \text{argmax } F(A) \text{ s.t. } A \subseteq V}$$



- Example:
  - F(A) = U(A) – C(A) where U(A) is submodular utility, and C(A) is supermodular cost function
    E.g.: Trading off utility and privacy in personalized search
    [Krause & Horvitz AAAI '08]

- In general: NP hard. Moreover:
- If F(A) can take negative values:
  As hard to approximate as maximum independent set
  (i.e., NP hard to get $O(n^{1-\varepsilon})$ approximation)

Maximizing positive submodular functions
[Feige, Mirrokni, Vondrak FOCS '07]

Theorem

There is an efficient randomized local search procedure, that, given a positive submodular function F, $F(\emptyset)=0$, returns set A_{LS} such that

$$F(A_{LS}) \geq (2/5) \max_A F(A)$$

- picking a random set gives ¼ approximation (½ approximation if F is symmetric!)
- we cannot get better than ¾ approximation unless P = NP

Scalarization vs. constrained maximization

Given monotonic utility F(A) and cost C(A), optimize:

Option 1:

$$\max_A F(A) - C(A)$$
$$\text{s.t. } A \subseteq V$$

"Scalarization"

Can get 2/5 approx…

if $F(A)-C(A) \geq 0$

for all $A \subseteq V$

Option 2:

$$\max_A F(A)$$
$$\text{s.t. } C(A) \leq B$$

"Constrained maximization"

coming up…

Positiveness is a strong requirement ☹

Constrained maximization: Outline

Monotonic submodular

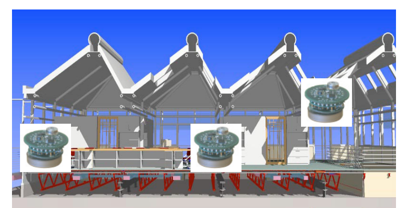
Selected set

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

Selection cost

Budget

$$\text{subject to } C(\mathcal{A}) \leq B$$



Subset selection: C(A) = |A|

Robust optimization

Complex constraints

Monotonicity

- A set function is called monotonic if

  $$A \subseteq B \subseteq V \Rightarrow F(A) \leq F(B)$$

- Examples:
  - Influence in social networks [Kempe et al KDD '03]
  - For discrete RVs, entropy $F(A) = H(X_A)$ is monotonic:
    Suppose $B = A \cup C$. Then
    $F(B) = H(X_A, X_C) = H(X_A) + H(X_C \mid X_A) \geq H(X_A) = F(A)$
  - Information gain: $F(A) = H(Y) - H(Y \mid X_A)$
  - Set cover
  - Matroid rank functions (dimension of vector spaces, …)
  - …

Subset selection

- Given: Finite set V, monotonic submodular function F, F(∅) = 0
- Want: $A^* \subseteq$ V such that

$$\boxed{\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}| \leq k} F(\mathcal{A})}$$

NP-hard!

Exact maximization of monotonic submodular functions

1) Mixed integer programming [Nemhauser et al '81]

$$\begin{aligned} &\max \eta \\ &\text{s.t.} \quad \eta \leq F(B) + \textstyle\sum_{s \in V \setminus B} \alpha_s \, \delta_s(B) \text{ for all } B \subseteq S \\ &\qquad \textstyle\sum_s \alpha_s \leq k \\ &\qquad \alpha_s \in \{0,1\} \end{aligned}$$

where $\delta_s(B) = F(B \cup \{s\}) - F(B)$

Solved using constraint generation

2) Branch-and-bound: "Data-correcting algorithm" [Goldengorin et al '99] M

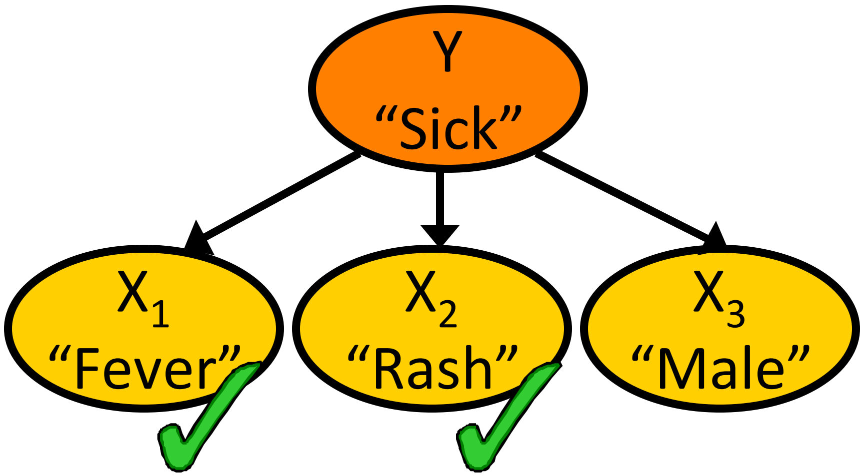
Both algorithms worst-case exponential!

Approximate maximization

- Given: finite set V, monotonic submodular function F(A)
  Want: $A^* \subseteq V$ such that

$$\boxed{\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}| \leq k} F(\mathcal{A})}$$

NP-hard!



<u>Greedy algorithm:</u> M

Start with $A_0 = \emptyset$

For i = 1 to k

$s_i := \operatorname{argmax}_s F(A_{i-1} \cup \{s\}) - F(A_{i-1})$

$A_i := A_{i-1} \cup \{s_i\}$

Performance of greedy algorithm

Theorem [Nemhauser et al '78]

Given a monotonic submodular function F, $F(\emptyset)=0$, the greedy maximization algorithm returns A_{greedy}

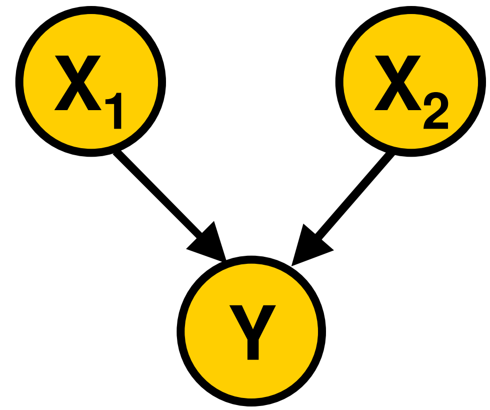
$$F(A_{greedy}) \geq \underbrace{(1-1/e)}_{\sim 63\%} \max_{|A| \leq k} F(A)$$

Sidenote: Greedy algorithm gives 1/2 approximation for maximization over any matroid C! [Fisher et al '78]

An "elementary" counterexample

$X_1, X_2 \sim$ Bernoulli(0.5)

$Y = X_1$ **XOR** X_2



Let $F(A) = IG(X_A; Y) = H(Y) - H(Y|X_A)$

$Y \mid X_1$ and $Y \mid X_2$ $\sim$ Bernoulli(0.5) (entropy 1)

$Y \mid X_1,X_2$ is deterministic! (entropy 0)

Hence $F(\{1,2\}) - F(\{1\}) = 1$, but

$F(\{2\}) - F(\emptyset) = 0$

F(A) submodular under some conditions! (later)

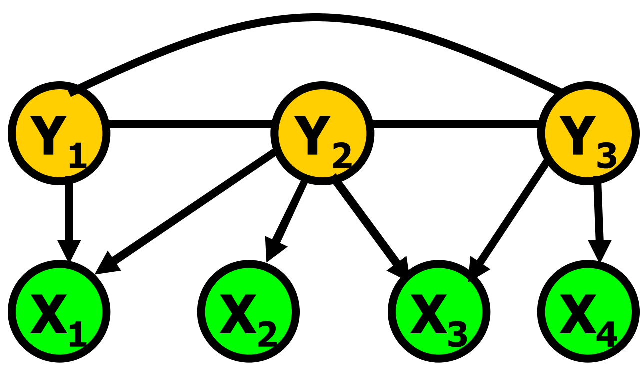
Example: Submodularity of info-gain

$Y_1,\dots,Y_m$, X_1, …, X_n discrete RVs

$F(A) = IG(Y; X_A) = H(Y)-H(Y \mid X_A)$

- F(A) is always monotonic
- However, NOT always submodular

Theorem [Krause & Guestrin UAI' 05]
If X_i are all conditionally independent given Y, then F(A) is submodular!



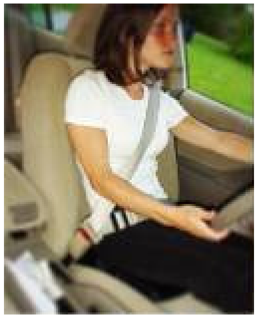
Hence, greedy algorithm works!

In fact, NO algorithm can do better than (1-1/e) approximation!

Building a Sensing Chair

[Mutlu, Krause, Forlizzi, Guestrin, Hodgins UIST '07]

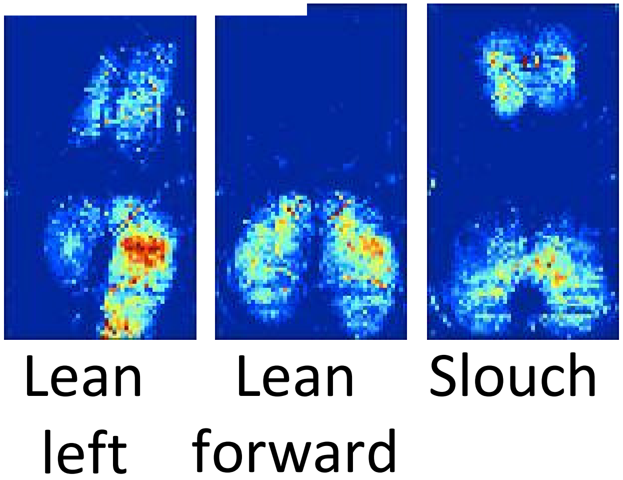
- People sit a lot
- Activity recognition in assistive technologies
- Seating pressure as user interface



Equipped with
1 sensor per cm^2!

Costs $16,000! ☹

Can we get similar accuracy with fewer, cheaper sensors?



82% accuracy on 10 postures! [Tan et al]

How to place sensors on a chair?

- Sensor readings at locations V as random variables
- Predict posture Y using probabilistic model P(Y,V)
- Pick sensor locations A* $\subseteq$ V to minimize entropy:

$$\boxed{\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}| \leq k} \mathrm{IG}(Y; X_{\mathcal{A}})}$$

Possible locations **V**

Placed sensors, did a user study:

	Accuracy	Cost
Before	82%	**$16,000** ☹
After		

Similar accuracy at <1% of the cost!

Variance reduction

(a.k.a. Orthogonal matching pursuit, Forward Regression)

- Let $Y = \sum_i \alpha_i X_i + \varepsilon$, and $(X_1,\ldots,X_n,\varepsilon) \sim N(\cdot;\mu,\Sigma)$
- Want to pick subset X_A to predict Y
- $Var(Y \mid X_A = x_A)$: conditional variance of Y given $X_A = x_A$
- Expected variance: $Var(Y \mid X_A) = \int p(x_A)\, Var(Y \mid X_A = x_A)\, dx_A$
- Variance reduction: $F_V(A) = Var(Y) - Var(Y \mid X_A)$

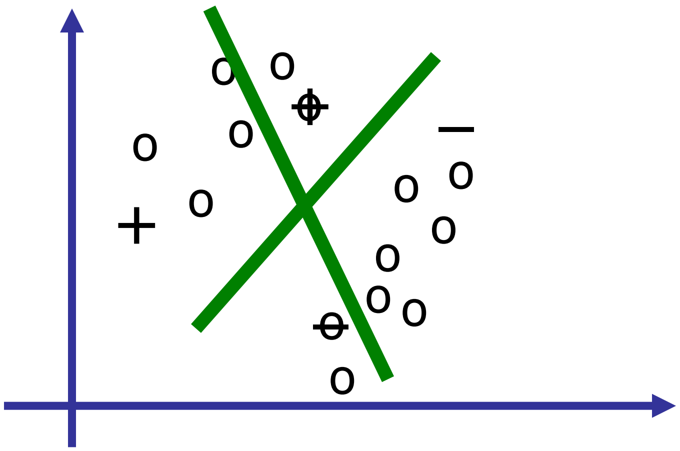
$F_V(A)$ is always monotonic

Theorem [Das & Kempe, STOC '08]
$F_V(A)$ is submodular*

*under some conditions on Σ

➔ **Orthogonal matching pursuit near optimal!**
[see other analyses by Tropp, Donoho et al., and Temlyakov]

Batch mode active learning [Hoi et al, ICML'06]



Which data points o should we label to minimize error?

Want batch A of k points to show an expert for labeling

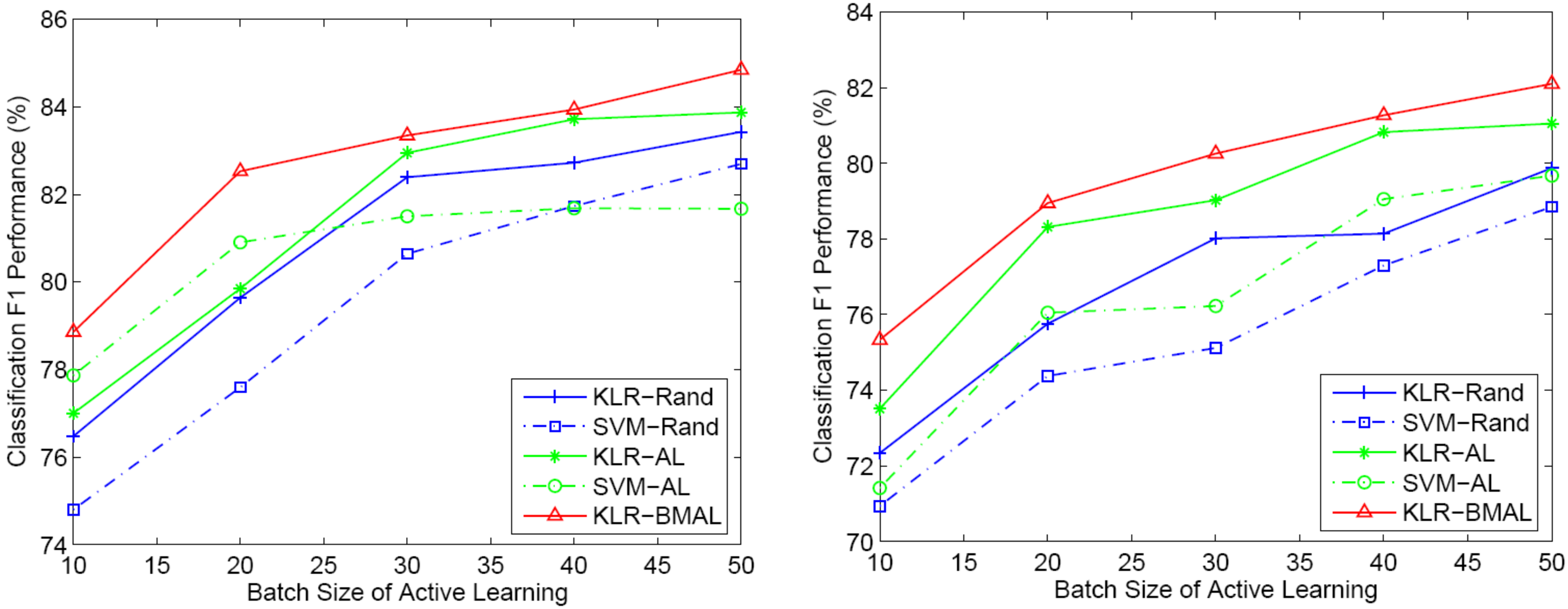
$$F(\mathcal{A}) = \frac{1}{\delta}\sum_{s\in\mathcal{V}}\sigma^2(s) - \sum_{s\notin\mathcal{A}}\frac{\sigma^2(s)}{\delta + \sum_{s'\in\mathcal{A}}\sigma^2(s')(s^T s')}$$

- F(A) selects examples that are
  - uncertain [$\sigma^2(s) = \pi(s)\,(1-\pi(s))$ is large]
  - diverse (points in A are as different as possible)
  - relevant (as close to $V\setminus A$ is possible, $s^T s'$ large)
- F(A) is submodular and monotonic!

  [approximation to improvement in Fisher-information]

Results about Active Learning

[Hoi et al, ICML'06]



(a) Australian (b) Heart

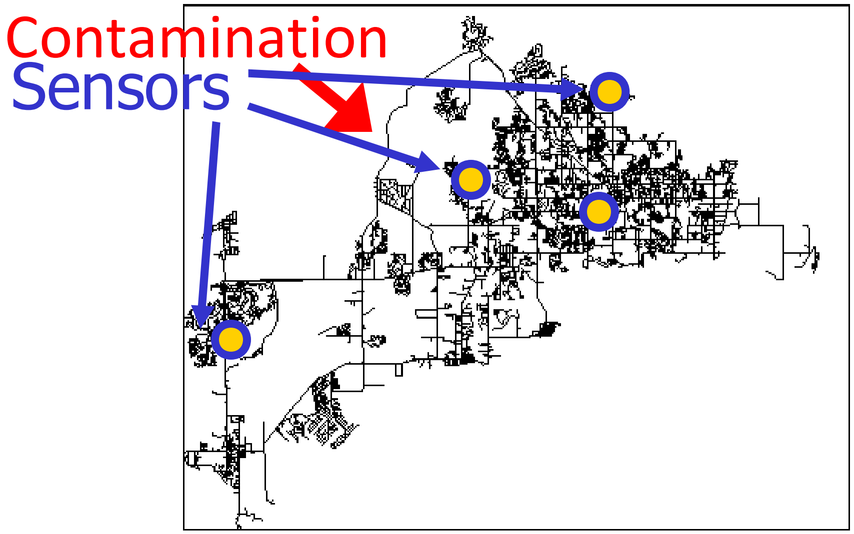
Batch mode Active Learning performs better than

- Picking k points at random
- Picking k points of highest entropy

Monitoring water networks

[Krause et al, J Wat Res Mgt 2008]

- Contamination of drinking water could affect millions of people



Simulator from EPA

Hach Sensor

~$14K

- Place sensors to detect contaminations
- "Battle of the Water Sensor Networks" competition

Where should we place sensors to quickly detect contamination?

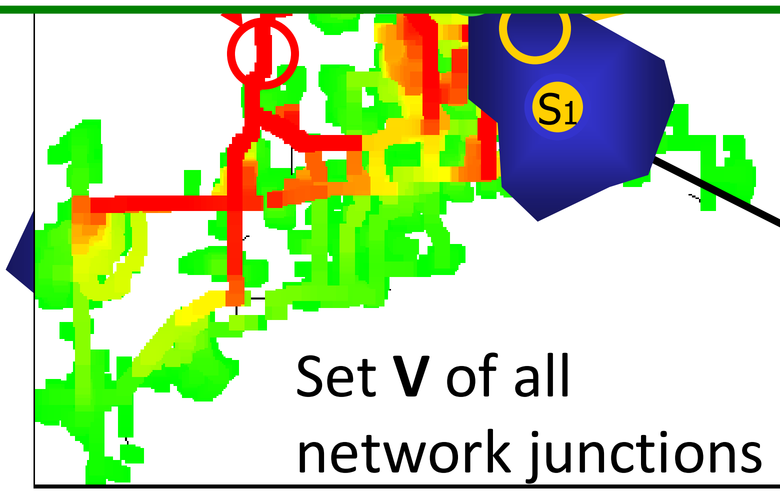
Model-based sensing

- Utility of placing sensors based on model of the world
  - For water networks: Water flow simulator from EPA
- F(A)=Expected impact reduction placing sensors at A

Model predicts Low impact

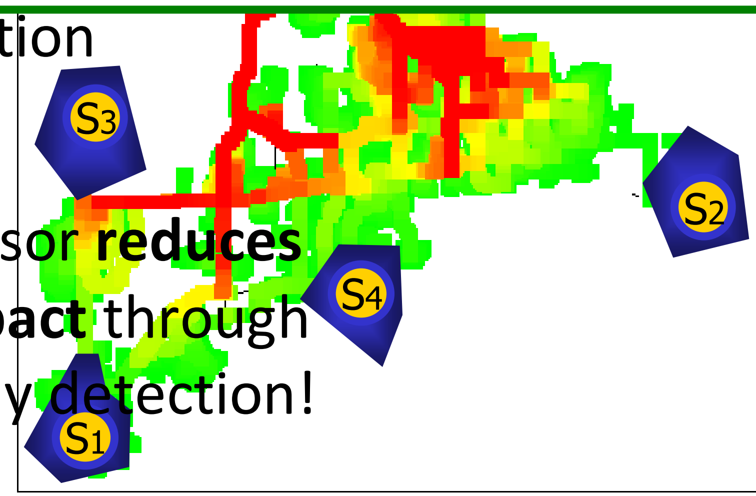
Theorem [Krause et al., J Wat Res Mgt '08]**:**

Impact reduction F(A) in water networks is submodular!



location

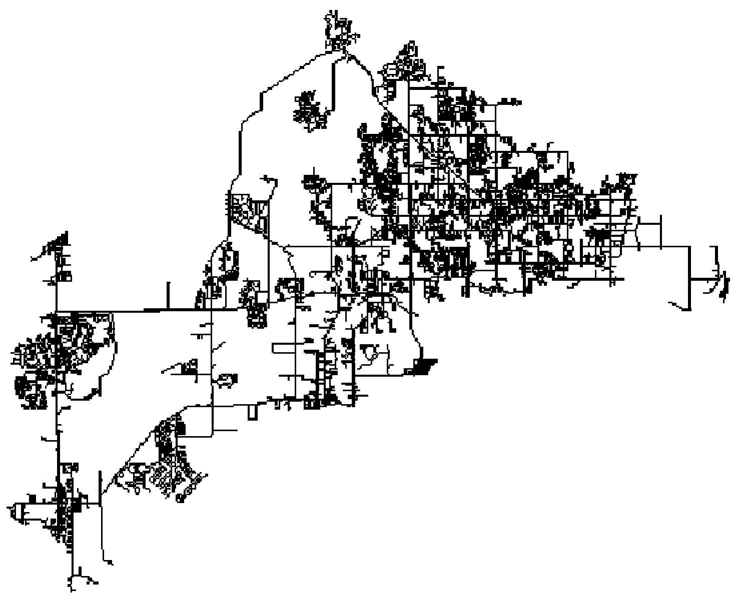
Sensor **reduces impact** through early detection!



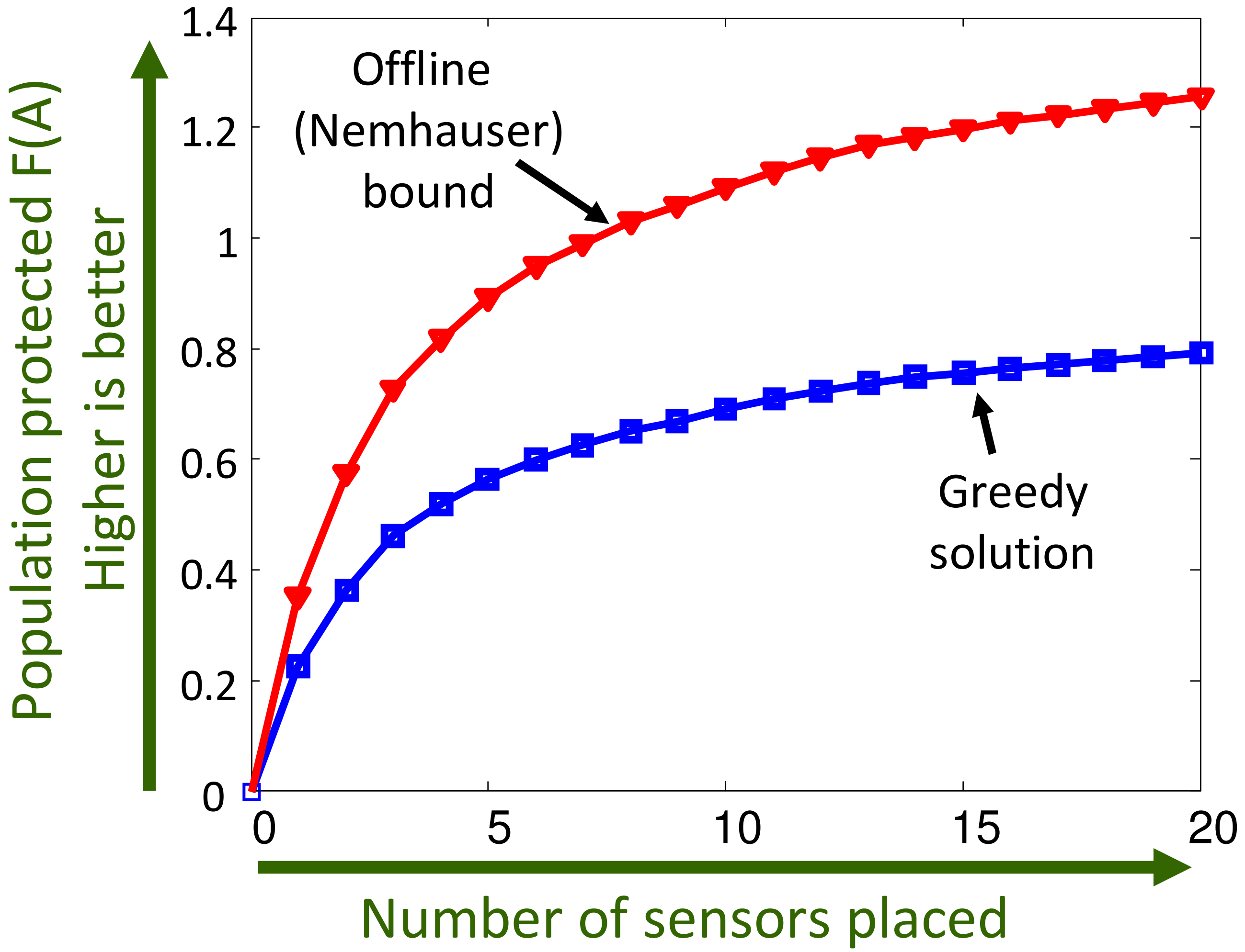
High impact reduction F(A) = 0.9 Low impact reduction F(A)=0.01

Battle of the Water Sensor Networks Competition

- Real metropolitan area network (12,527 nodes)
- Water flow simulator provided by EPA
- 3.6 million contamination events
- Multiple objectives:
  - Detection time, affected population, …
- Place sensors that detect well "on average"



Bounds on optimal solution
[Krause et al., J Wat Res Mgt '08]



Water networks data

(1-1/e) bound quite loose… can we get better bounds?

Data dependent bounds [Minoux '78]

- Suppose A is candidate solution to

$$\boxed{\operatorname{argmax} F(A) \;\; \text{s.t.} \; |A| \leq k}$$

and $A^* = \{s_1,\ldots,s_k\}$ be an optimal solution

- Then $F(A^*) \leq F(A \cup A^*)$

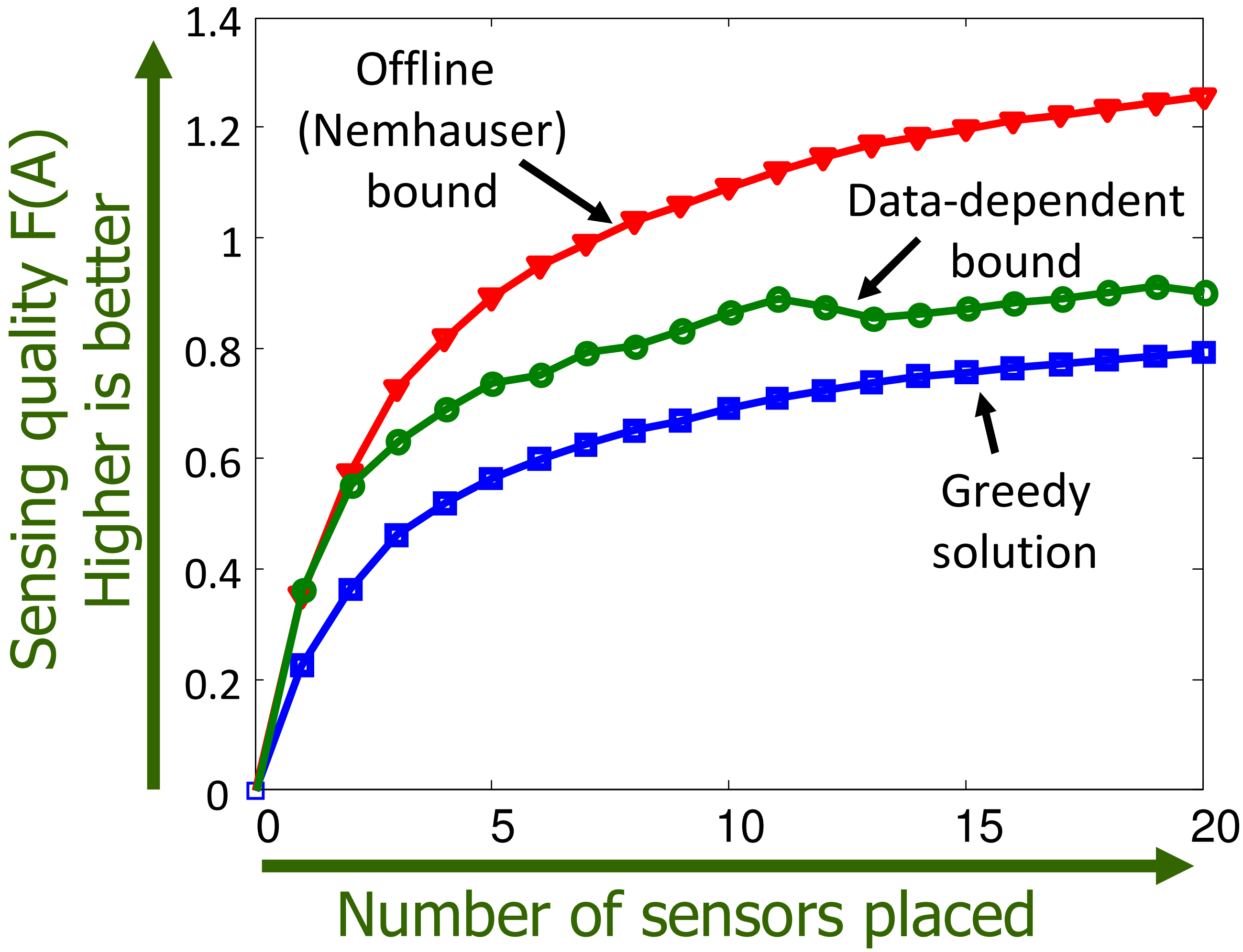
$$= F(A)+\sum_i F(A\cup\{s_1,\ldots,s_i\})-F(A\cup \{s_1,\ldots,s_{i-1}\})$$
$$\leq F(A) + \sum_i (F(A\cup\{s_i\})-F(A))$$
$$= F(A) + \sum_i \delta_{s_i}$$

For each $s \in V\backslash A$, let $\delta_s = F(A\cup\{s\})-F(A)$ $\quad M$

Order such that $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_n$

Then: $F(A^*) \leq F(A) + \sum_{i=1}^{k} \delta_i$

Bounds on optimal solution
[Krause et al., J Wat Res Mgt '08]



Water networks data

Submodularity gives **data-dependent** bounds on the performance of **any** algorithm

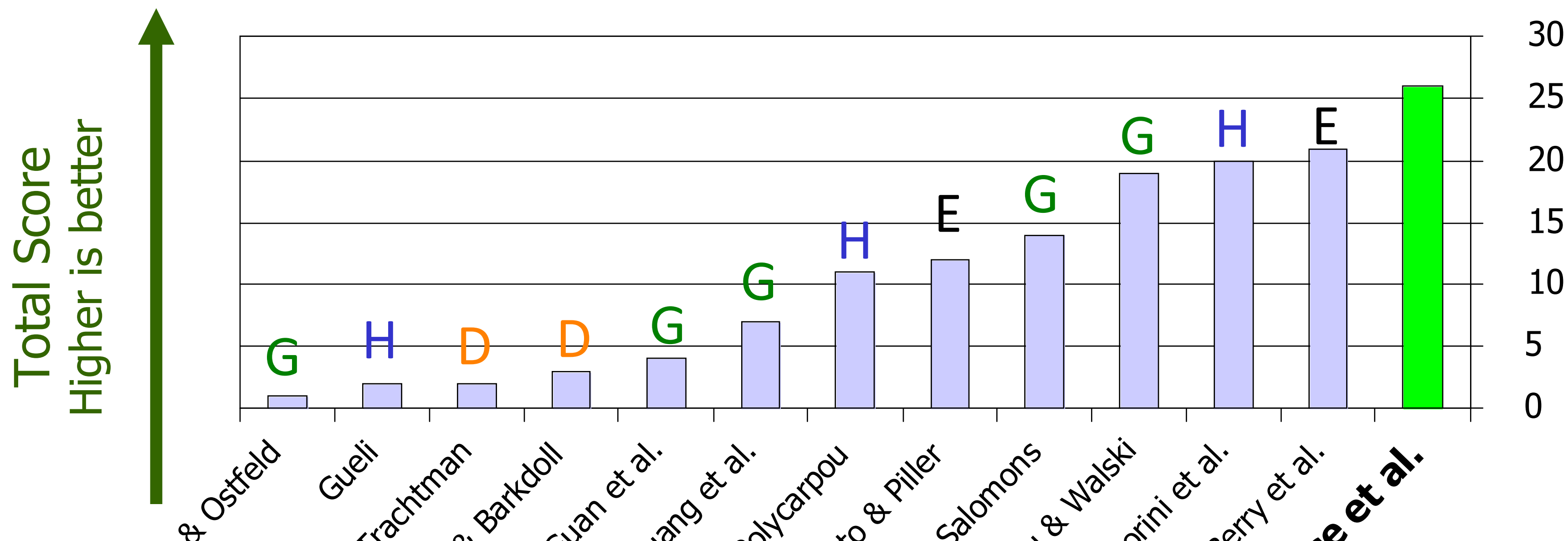
BWSN Competition results

[Ostfeld et al., J Wat Res Mgt 2008]

- 13 participants
- Performance measured in 30 different criteria

G: Genetic algorithm D: Domain knowledge

H: Other heuristic E: "Exact" method (MIP)



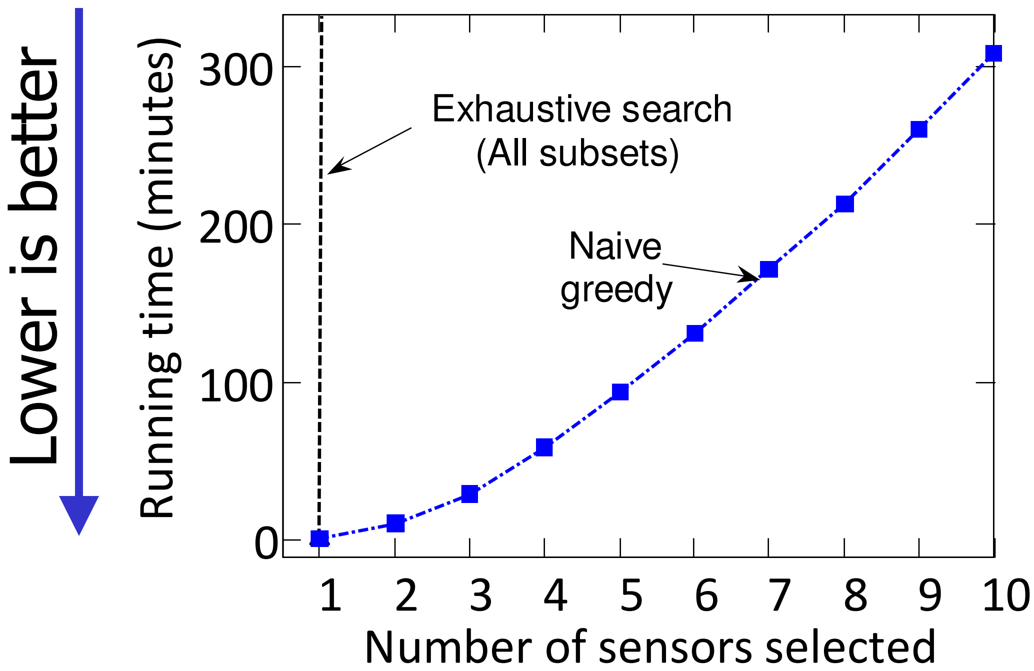
24% better performance than runner-up! ☺

What was the trick?

Simulated all **3.6M contaminations** on 2 weeks / 40 processors

152 GB data on disk, 16 GB in main memory (compressed)

➔ Very accurate computation of F(A)   Very slow evaluation of F(A) ☹



30 hours/20 sensors

6 weeks for all

30 settings ☹

Submodularity to the rescue

Scaling up greedy algorithm [Minoux '78]

In round i+1,

- have picked $A_i = \{s_1,\ldots,s_i\}$
- pick $s_{i+1} = \text{argmax}_s\ F(A_i \cup \{s\})-F(A_i)$

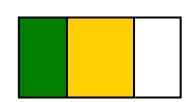
I.e., maximize "marginal benefit" $\delta_s(A_i)$

$$\delta_s(A_i) = F(A_i \cup \{s\})-F(A_i)$$

Key observation: Submodularity implies

$$i \leq j \Rightarrow \delta_s(A_i) \geq \delta_s(A_j)$$

$\delta_s(A_i) \geq \delta_s(A_{i+1})$

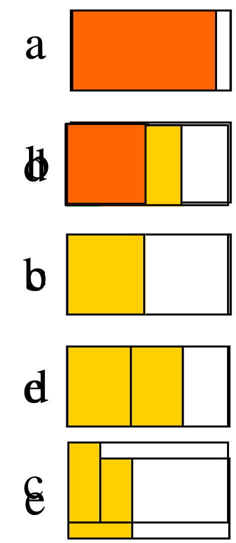
s 

Marginal benefits can never increase!

"Lazy" greedy algorithm
[Minoux '78]

Lazy greedy algorithm: M

- First iteration as usual
- Keep an ordered list of marginal benefits δ_i from previous iteration
- Re-evaluate δ_i only for top element
- If δ_i stays on top, use it, otherwise re-sort



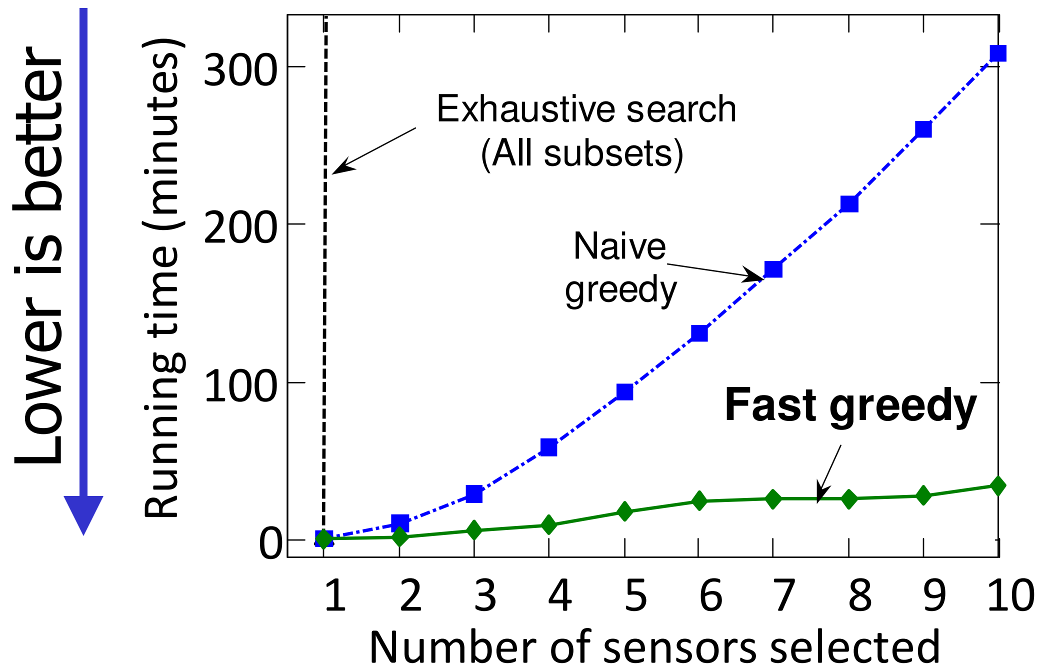
Note: Very easy to compute online bounds, lazy evaluations, etc. [Leskovec et al. '07]

Result of lazy evaluation

Simulated all **3.6M contaminations** on 2 weeks / 40 processors

152 GB data on disk, 16 GB in main memory (compressed)

➔ Very accurate computation of F(A) Very slow evaluation of F(A) ☹



30 hours/20 sensors

6 weeks for all
30 settings ☹

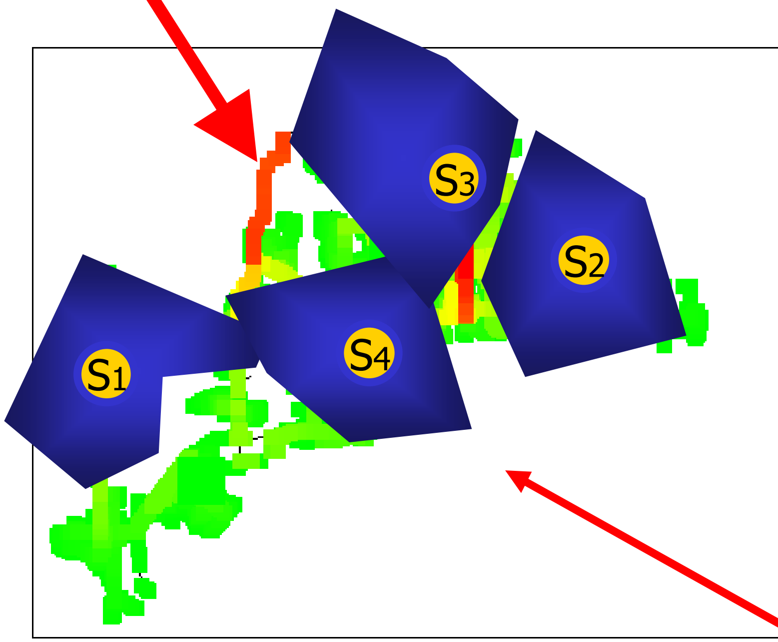
Submodularity to the rescue:

Using "lazy evaluations":
1 hour/20 sensors

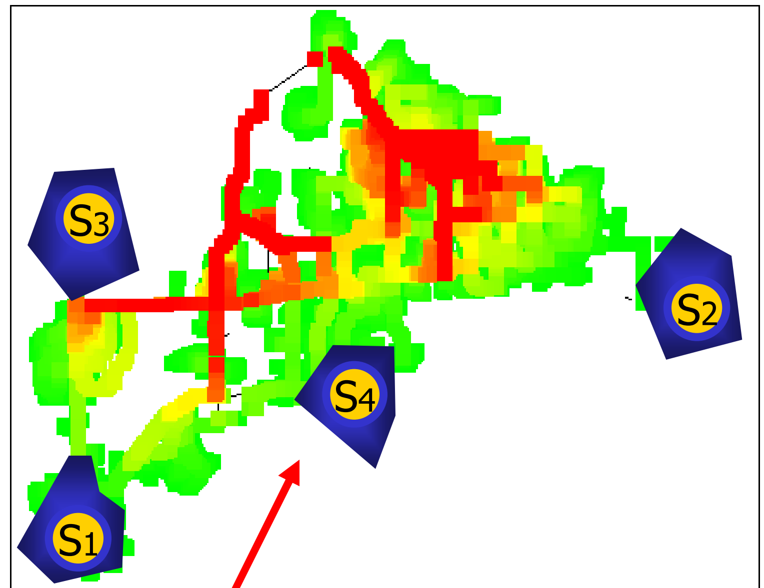
Done after 2 days! ☺

What about worst-case?
[Krause et al., NIPS ’07]

Knowing the sensor locations, an adversary contaminates here!



Placement detects well on **“average-case”** (accidental) contamination



Very different average-case impact, **Same** worst-case impact

Where should we place sensors to quickly detect in the **worst case**?

Constrained maximization: Outline

Utility function

Selected set

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

Selection cost

Budget

$$\text{subject to } C(\mathcal{A}) \leq B$$

Subset selection ✓

Robust optimization

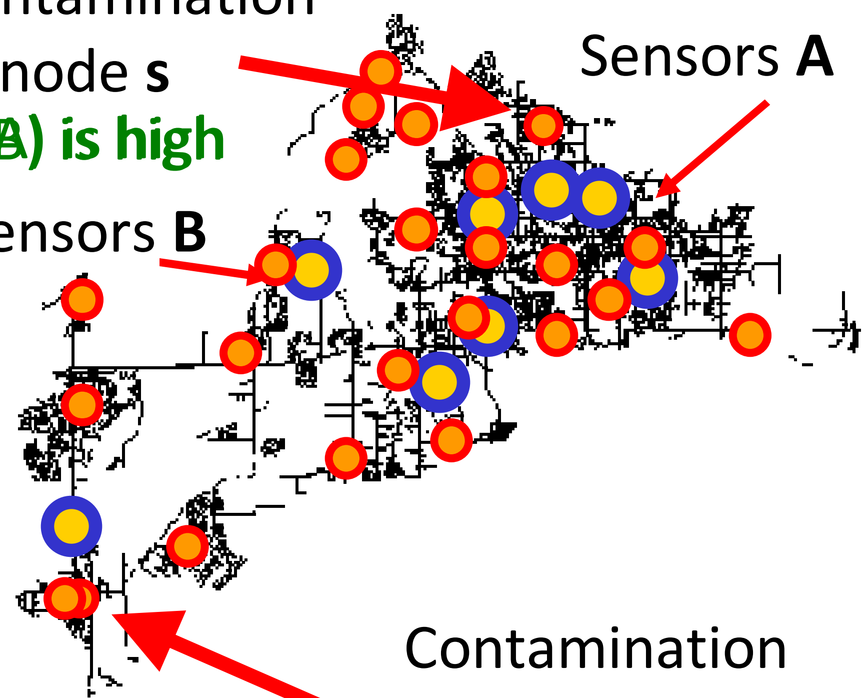
Complex constraints

Optimizing for the worst case

- Separate utility function F_i for each contamination i
- F_i(A) = impact reduction by sensors A for contamination i

Want to solve

$$\boxed{\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}|\leq k} \min_i F_i(\mathcal{A})}$$



Each of the F_i is submodular

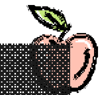
Unfortunately, $\min_i F_i$ not submodular!

How can we solve this **robust optimization** problem?

How does the greedy algorithm do?

V={  }

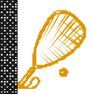
Can only buy k=2

Greedy picks  first

Optimal solution

Hence we can't find **any** approximation algorithm.

Or can we?

Then, can choose only  or 

Optimal score: 1

Greedy score: ε

➔ Greedy does arbitrarily badly. Is there something better?

Theorem [NIPS '07]**:** The problem $\max_{|A| \leq k} \min_i F_i(A)$ does not admit **any** approximation unless **P=NP**

Alternative formulation

If somebody told us the optimal value,

$$c^* = \max_{|\mathcal{A}| \leq k} \min_i F_i(\mathcal{A})$$

can we recover the optimal solution A^*?

Need to find

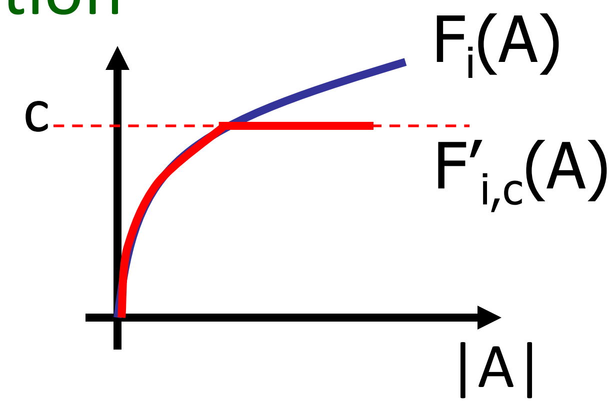
$$\mathcal{A}^* = \operatorname*{argmin}_{\mathcal{A}} |\mathcal{A}| \text{ such that } \min_i F_i(\mathcal{A}) \geq c^*$$

Is this any easier?

Yes, if we relax the constraint $|A| \leq k$

Solving the alternative problem

Trick: For each F_i and c, define truncation



Remains submodular!

$$F'_{i,c}(\mathcal{A}) = \min\{F_i(\mathcal{A}), c\}$$

$$F'_{\text{avg},c}(\mathcal{A}) = \frac{1}{m}\sum_i F'_{i,c}(\mathcal{A})$$

Problem 1 (last slide)

$$\min_{\mathcal{A}} |\mathcal{A}| \quad \text{s.t.} \quad \boxed{\min_i F_i(\mathcal{A})} \geq c$$

Problem 2

$$\min_{\mathcal{A}} |\mathcal{A}| \quad \text{s.t.} \quad \boxed{F'_{\text{avg},c}(\mathcal{A})} \geq c$$

Non-submodular ☹

Don't know how to solve

Same optimal solutions!

Solving one solves the other

Submodular!

But appears as constraint?

Maximization vs. coverage

Previously: Wanted

$$A^* = \text{argmax } F(A) \;\text{ s.t. } \; |A| \leq k$$

Now need to solve:

$$A^* = \text{argmin } |A| \;\text{ s.t. } \; F(A) \geq Q$$

<u>Greedy algorithm:</u>

M

Start with $A := \emptyset$;

While $F(A) < Q$ and $|A| < n$

$s^* := \text{argmax}_s F(A \cup \{s\})$

$A := A \cup \{s^*\}$

For bound, assume F is integral.
If not, just round it.

Theorem [Wolsey et al]: Greedy will return A_{greedy}

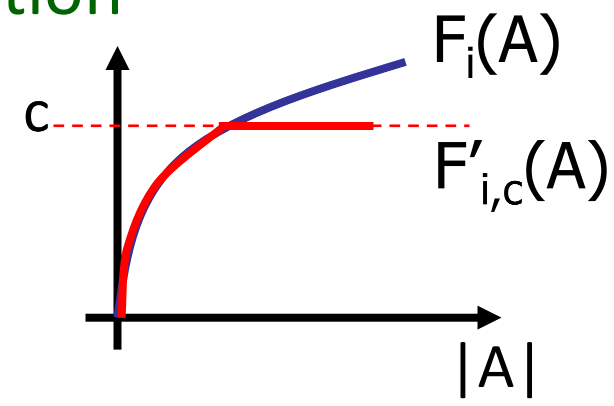
$|A_{greedy}| \leq (1+\log \max_s F(\{s\})) \; |A_{opt}|$

Solving the alternative problem

Trick: For each F_i and c, define truncation

$$F'_{i,c}(\mathcal{A}) = \min\{F_i(\mathcal{A}), c\}$$

$$F'_{\text{avg},c}(\mathcal{A}) = \frac{1}{m}\sum_i F'_{i,c}(\mathcal{A})$$



Problem 1 (last slide)

$$\min_{\mathcal{A}} |\mathcal{A}|$$
$$\text{s.t. } \boxed{\min_i F_i(\mathcal{A})} \geq c$$

Non-submodular ☹
Don't know how to solve

Problem 2

$$\min_{\mathcal{A}} |\mathcal{A}|$$
$$\text{s.t. } \boxed{F'_{\text{avg},c}(\mathcal{A})} \geq c$$

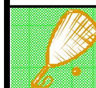
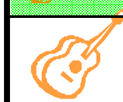
Submodular!
Can use greedy algorithm!

Back to our example

- Guess c=1
- First pick 
- Then pick 

➔ Optimal solution!

Set A	F_1	F_2	$\min_i F_i$
	1	0	0
	0	2	0
	ε	ε	ε

How do we find c?

Do binary search!

SATURATE Algorithm

[Krause et al, NIPS '07]

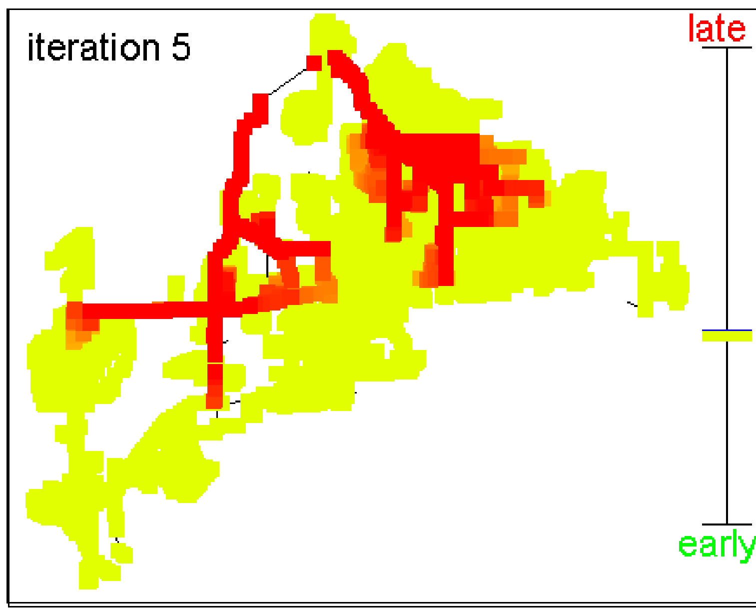
Given: set V, integer k and monotonic SFs $F_1,\ldots,F_m$

Initialize $c_{min}=0$, $c_{max} = \min_i F_i(V)$

Do binary search: $c = (c_{min}+c_{max})/2$

- Greedily find A_G such that $F'_{avg,c}(A_G) = c$
- If $|A_G| \leq \alpha$ k: increase c_{min}
- If $|A_G| > \alpha$ k: decrease c_{max}

until convergence



Truncation threshold (color)

Theoretical guarantees

[Krause et al, NIPS '07]

Theorem: The problem $\max_{|A| \leq k} \min_i F_i(A)$ does not admit **any** approximation unless **P=NP** ☹

Theorem: *SATURATE* finds a solution A_S such that

$$\min_i F_i(A_S) \geq OPT_k \text{ and } |A_S| \leq \alpha k$$

where $OPT_k = \max_{|A| \leq k} \min_i F_i(A)$

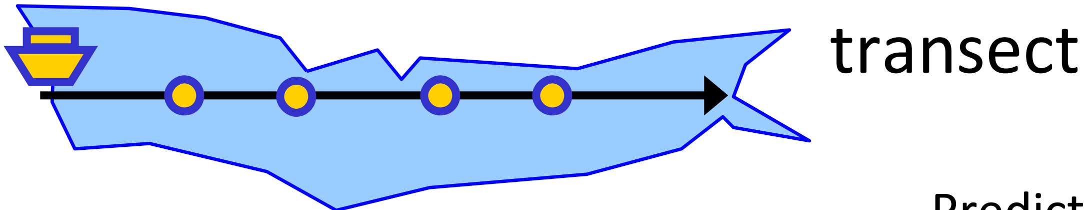
$\alpha = 1 + \log \max_s \sum_i F_i(\{s\})$

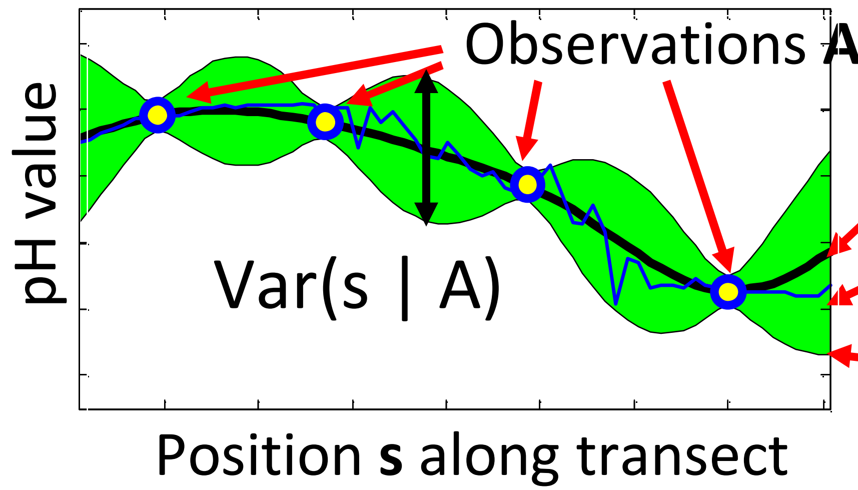
Theorem:

If there were a polytime algorithm with better factor $\beta < \alpha$, then $NP \subseteq DTIME(n^{\log \log n})$

Example: Lake monitoring

- Monitor pH values using robotic sensor





Prediction at unobserved locations

True (hidden) pH values

Use **probabilistic model** (Gaussian processes) to estimate prediction error

Where should we sense to **minimize our maximum error**?

$$\min_s \underbrace{\mathrm{Var}(s) - \mathrm{Var}(s \mid \mathcal{A})}_{\text{(often) submodular}}$$

[Das & Kempe '08]

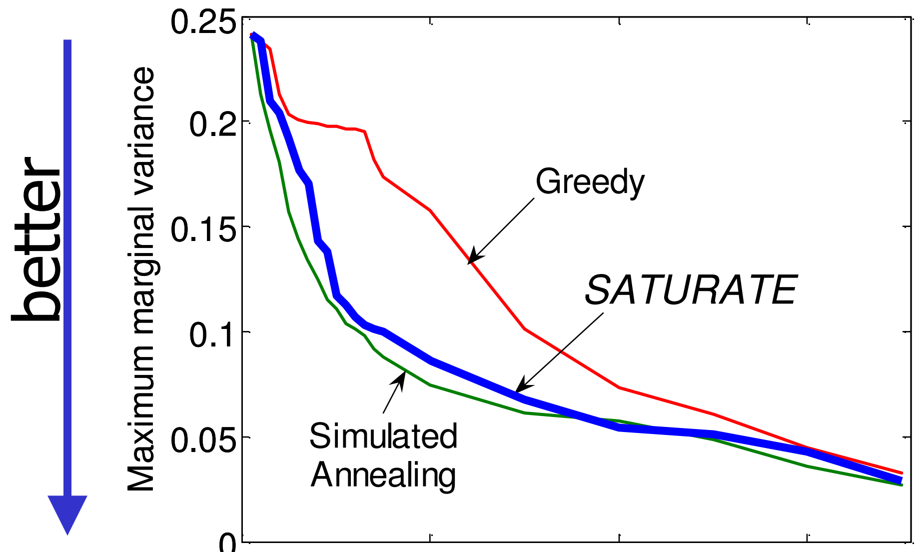
➔ Robust submodular optimization problem!

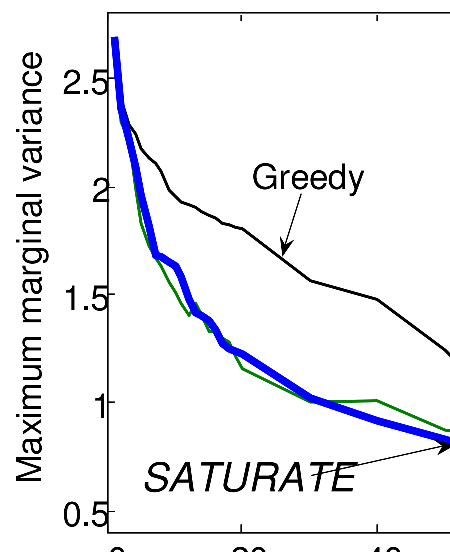
Comparison with state of the art

Algorithm used in geostatistics: *Simulated Annealing*

[Sacks & Schiller ’88, van Groeningen & Stein ’98, Wiens ’05,…]

7 parameters that need to be fine-tuned





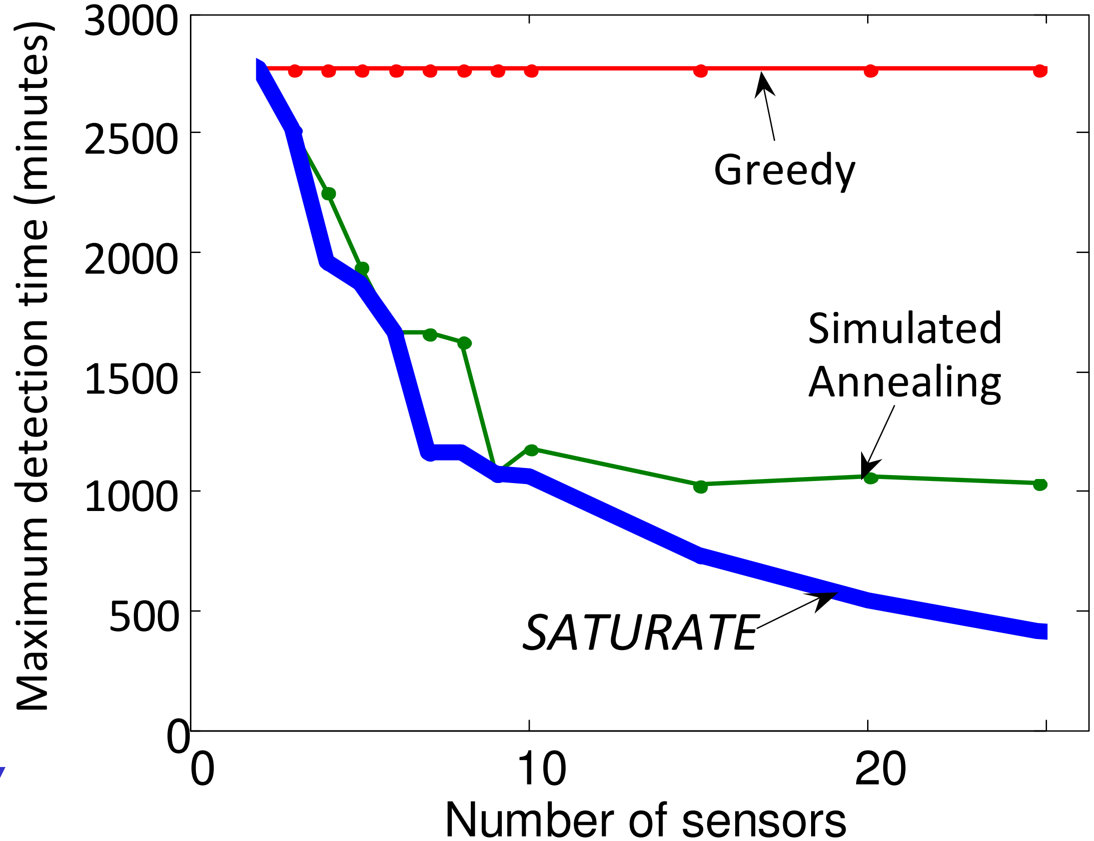
Envi

SATURATE is competitive & 10x faster

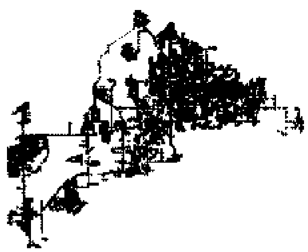
No parameters to tune!

Results on water networks

Lower is better



No decrease until **all** contaminations detected!



Water networks

60% lower worst-case detection time!

Worst- vs. average case

Given: Set V, submodular functions $F_1,\ldots,F_m$

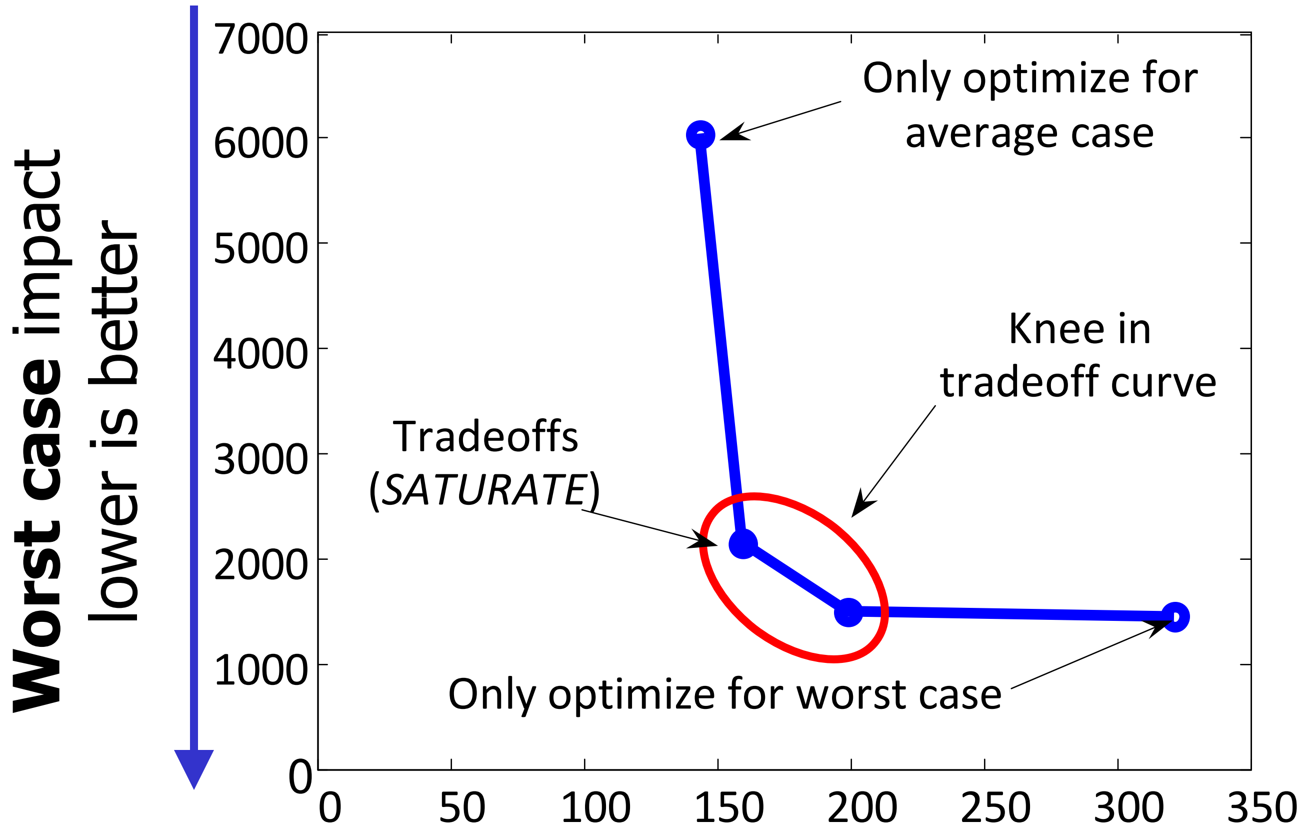
Average-case score	Worst-case score
$F_{ac}(\mathcal{A}) = \frac{1}{m}\sum_i F_i(\mathcal{A})$	$F_{wc}(\mathcal{A}) = \min_i F_i(\mathcal{A})$

Want to optimize **both** average- and worst-case score!

Can modify *SATURATE* to solve this problem! ☺

- Want: $F_{ac}(A) \geq c_{ac}$ and $F_{wc}(A) \geq c_{wc}$
- Truncate: $\min\{F_{ac}(A),c_{ac}\} + \min\{F_{wc}(A),c_{wc}\} \geq c_{ac}+c_{wc}$

Worst- vs. average case



Water networks data

Can find good compromise between average- and worst-case score!

Constrained maximization: Outline

Utility function

Selected set

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

Selection cost

Budget

$$\text{subject to } C(\mathcal{A}) \leq B$$

Subset selection ✓

Robust optimization ✓

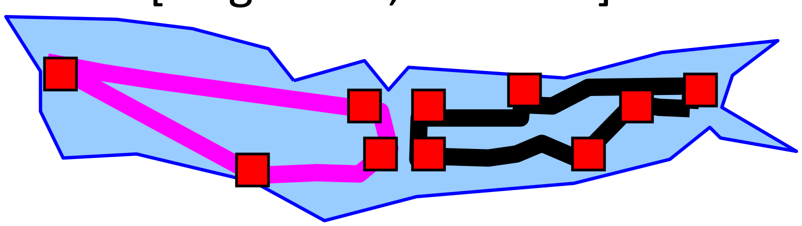
Complex constraints

Other aspects: Complex constraints

$\max_{\mathbf{A}} F(\mathbf{A})$ or $\max_{\mathbf{A}} \min_i F_i(\mathbf{A})$ subject to

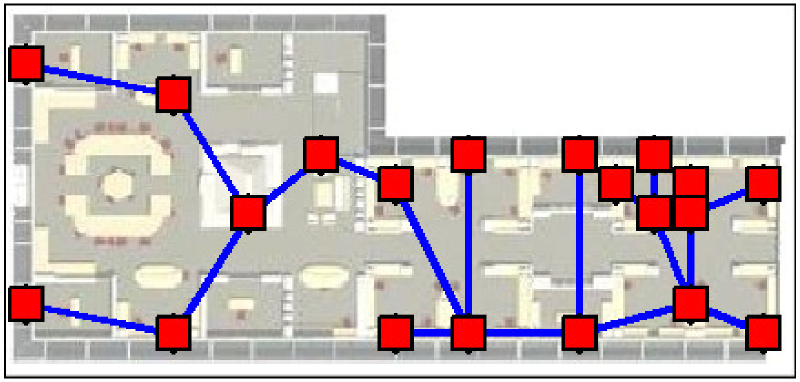
- So far: $|\mathbf{A}| \leq k$
- In practice, more complex constraints:
- Different costs: $C(A) \leq B$

Locations need to be connected by paths
[Chekuri & Pal, FOCS '05]
[Singh et al, IJCAI '07]



Lake monitoring

Sensors need to communicate (form a routing tree)



Building monitoring

Non-constant cost functions

- For each $s \in V$, let $c(s)>0$ be its cost (e.g., feature acquisition costs, …)
- Cost of a set $C(A) = \sum_{s\in A} c(s)$ (modular function!)
- Want to solve

$$A^* = \operatorname{argmax} F(A) \text{ s.t. } C(A) \leq B$$

M

<u>Cost-benefit greedy algorithm:</u>

Start with $A := \emptyset$;

While there is an $s\in V\setminus A$ s.t. $C(A\cup\{s\}) \leq B$

$$s^* = \underset{s:C(\mathcal{A}\cup\{s\})\leq B}{\operatorname{argmax}} \frac{F(\mathcal{A}\cup\{s\}) - F(\mathcal{A})}{c(s)}$$

$A := A \cup \{s^*\}$

Performance of cost-benefit greedy

Want

$\max_A F(A)$ s.t. $C(A) \leq 1$

Set A	F(A)	C(A)
{a}	2ε	ε
{b}	1	1

Cost-benefit greedy picks a.

Then cannot afford b!

➔ Cost-benefit greedy performs arbitrarily badly!

Cost-benefit optimization [Wolsey '82, Sviridenko '04, Leskovec et al '07]

Theorem [Leskovec et al. KDD '07]

- A_{CB}: cost-benefit greedy solution and
- A_{UC}: unit-cost greedy solution (i.e., ignore costs)

Then

$$\max \{ F(A_{CB}), F(A_{UC}) \} \geq ½ (1-1/e) \text{ OPT}$$

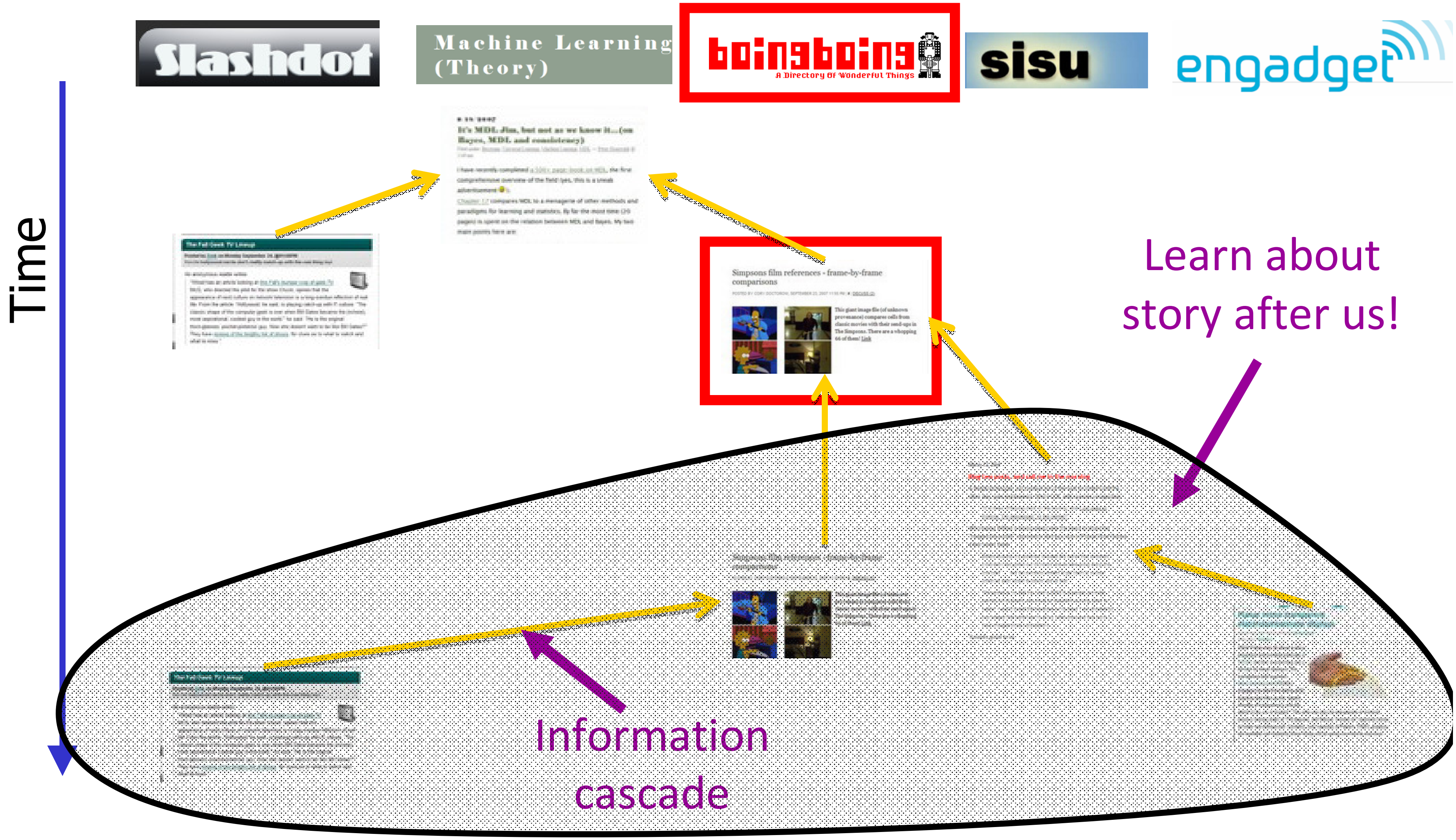
Can still compute online bounds and speed up using lazy evaluations

Note: Can also get

- (1-1/e) approximation in time $O(n^4)$ [Sviridenko '04]
- Slightly better than ½ (1-1/e) in $O(n^2)$ [Wolsey '82]

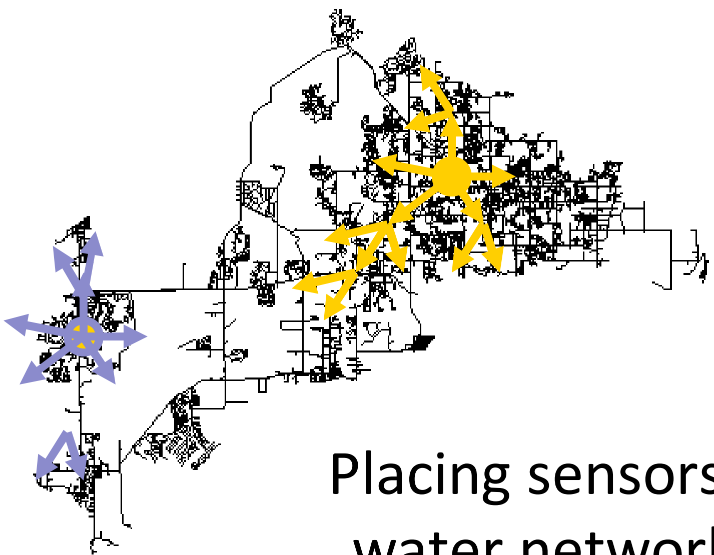
Example: Cascades in the Blogosphere

[Leskovec, Krause, Guestrin, Faloutsos, VanBriesen, Glance ‘07]



Which blogs should we read to learn about big cascades early?

Water vs. Web



Placing sensors in water networks

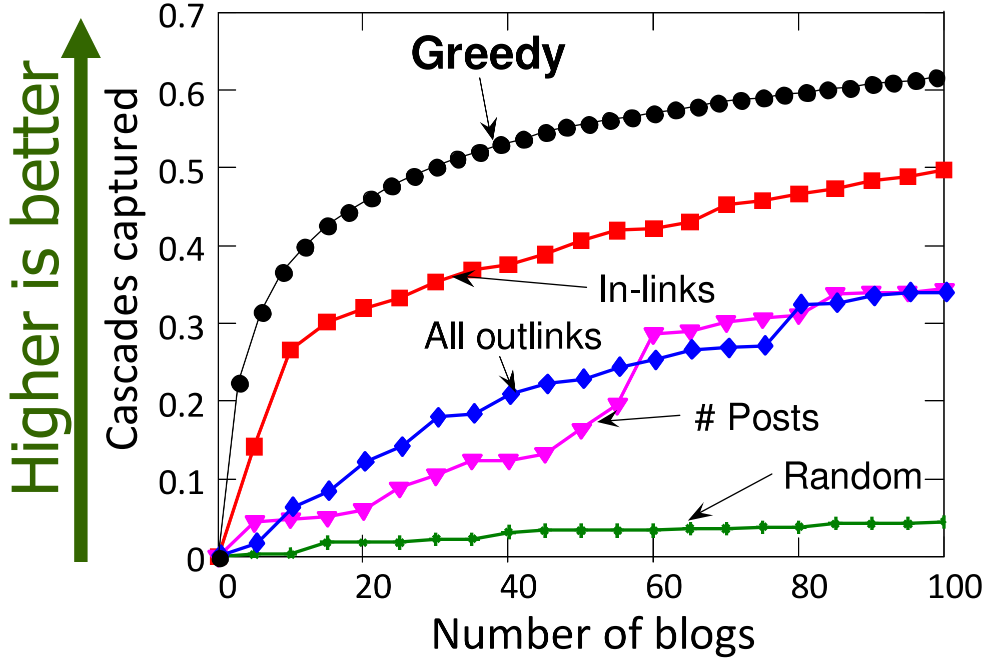
vs.

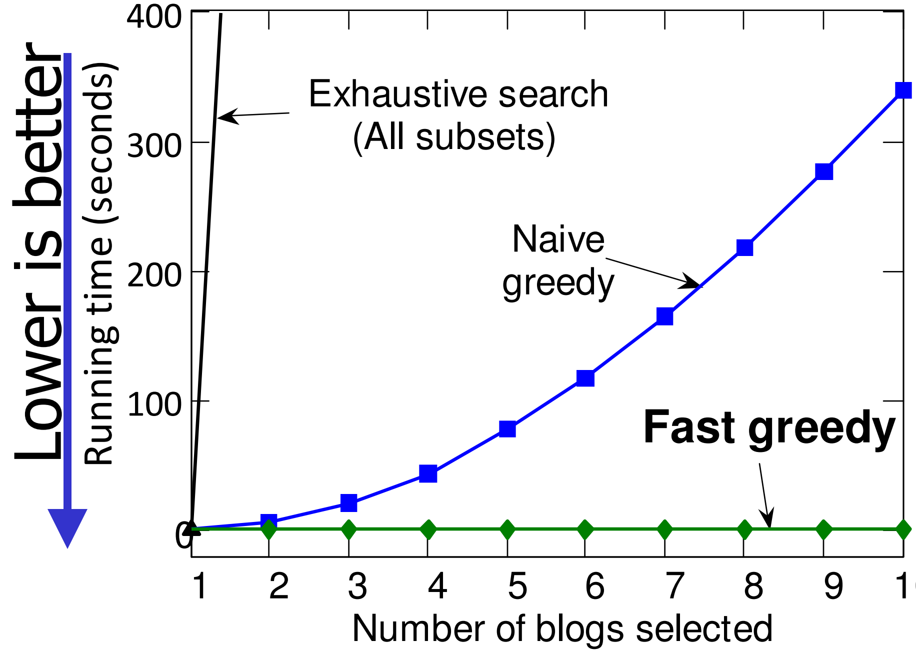
Selecting informative blogs

- In both problems we are given
  - Graph with nodes (junctions / blogs) and edges (pipes / links)
  - Cascades spreading dynamically over the graph (contamination / citations)
- Want to pick nodes to **detect big cascades early**

In both applications, utility functions submodular ☺
[Generalizes Kempe et al, KDD ’03]

Performance on Blog selection





Blog selection

~45k blogs

Blog selection

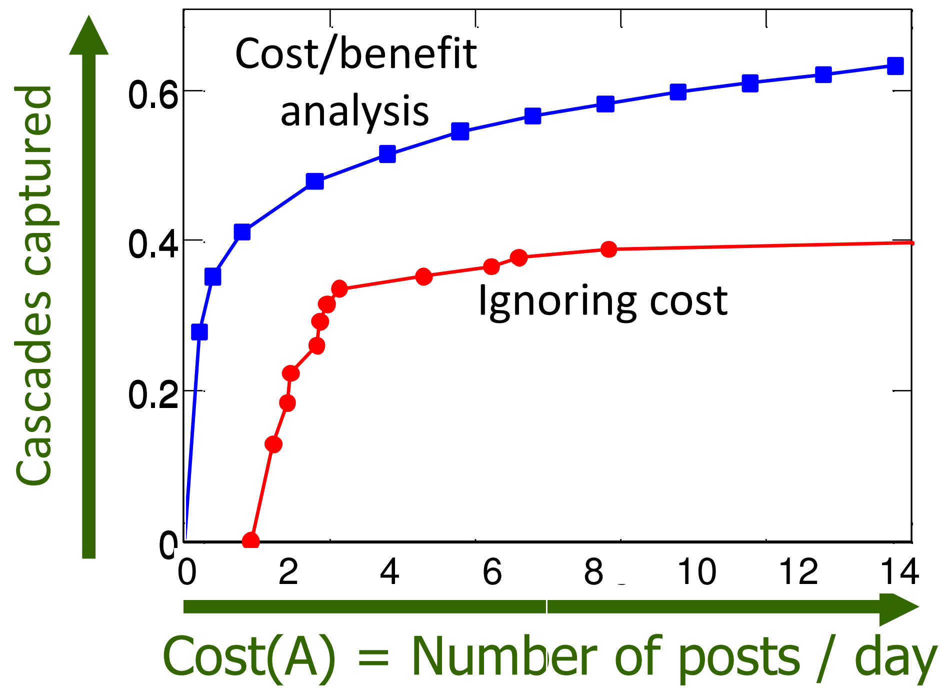
Outperforms state-of-the-art heuristics

700x speedup using submodularity!

Cost of reading a blog

skip

- Naïve approach: Just pick 10 best blogs
- Selects big, well known blogs (Instapundit, etc.)
- These contain many posts, take long to read!

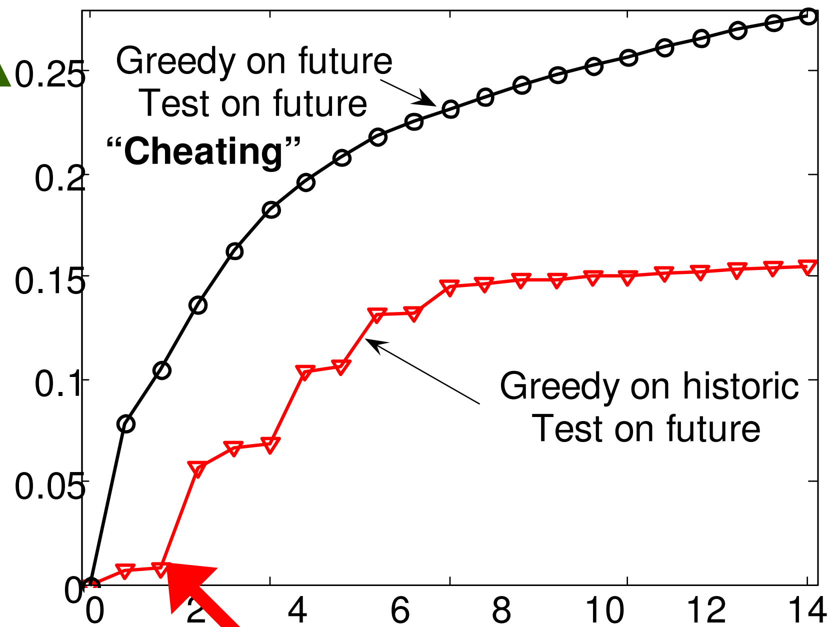


Cost-benefit optimization picks summarizer blogs!

Predicting the “hot” blogs

- Want blogs that will be informative in the future
- Split data set; train on historic, test on future

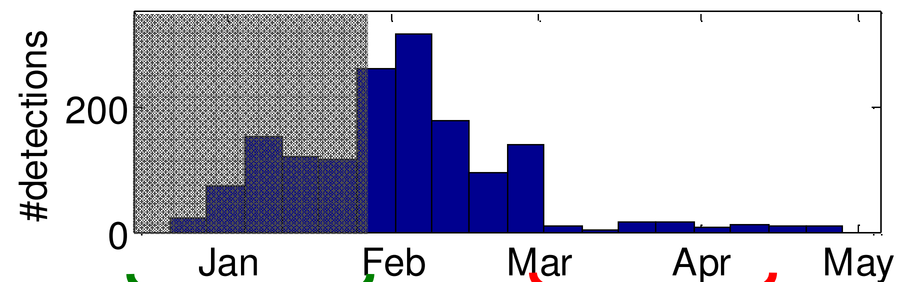
Cascades captured



Cost(A) = Number of posts / day

Let’s see what goes wrong here.

Detects on training set



Detect well here!

Detect poorly here!

Blog selection “overfits” to training data!

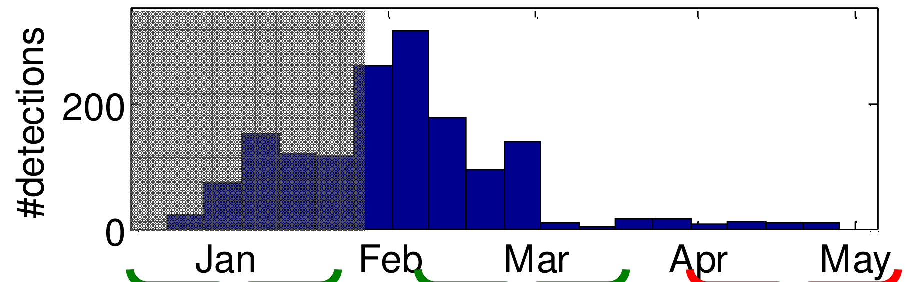
Poor generalization!

Why’s that?

Want blogs that continue to do well!

Robust optimization

"Overfit" blog selection **A**



$F_i(A)$ = detections in interval i

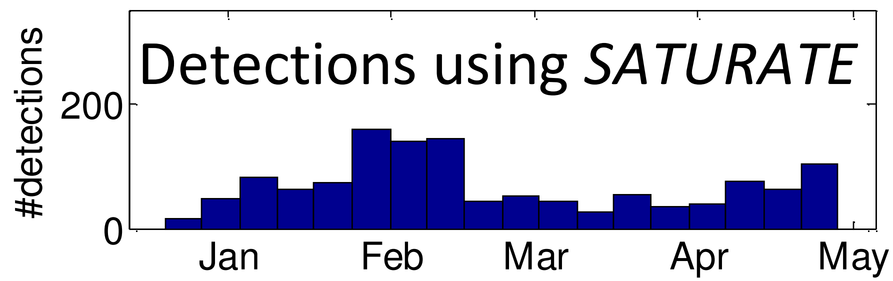
$F_1(A)$=.5 $F_3(A)$=.6 $F_5(A)$=.02

$F_2(A)$=.8 $F_4(A)$=.01

Optimize worst-case

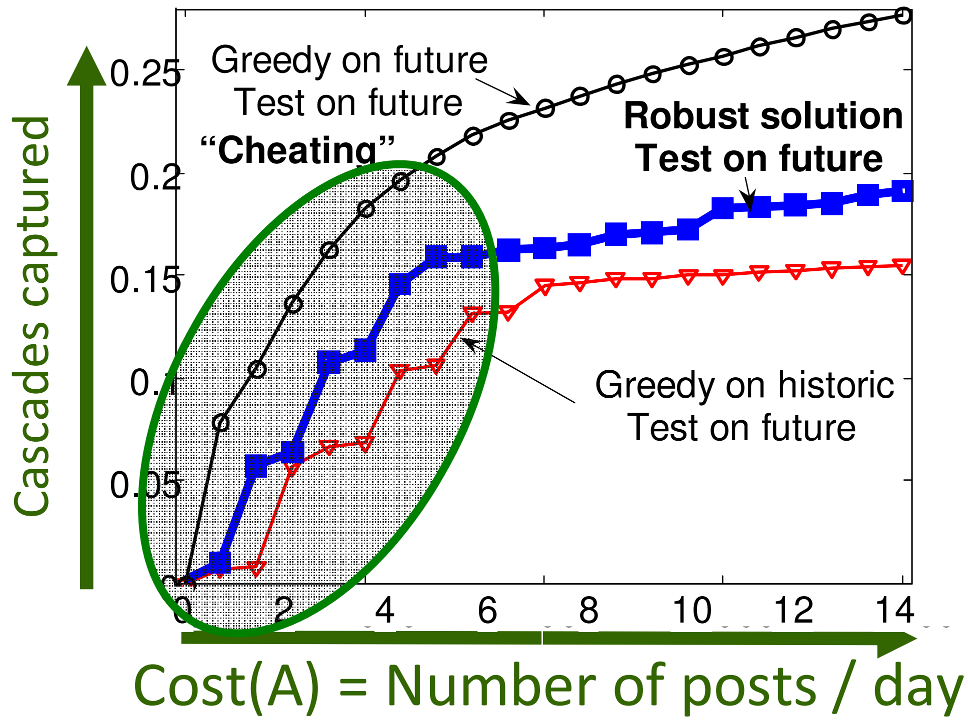
$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}|\le k} \min_i F_i(\mathcal{A})$$

"Robust" blog selection **A***



Robust optimization ⇔ Regularization!

Predicting the "hot" blogs



50% better generalization!

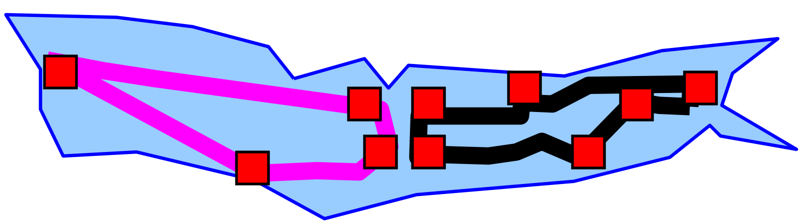
Other aspects: Complex constraints

skip

$\max_{\mathbf{A}} F(\mathbf{A})$ or $\max_{\mathbf{A}} \min_i F_i(\mathbf{A})$ subject to

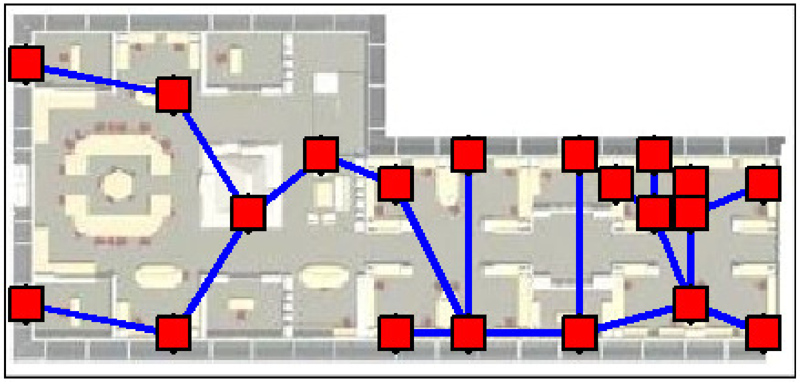
- So far: $|\mathbf{A}| \leq k$
- In practice, more complex constraints:
- Different costs: $C(A) \leq B$

Locations need to be connected by paths
[Chekuri & Pal, FOCS '05]
[Singh et al, IJCAI '07]



Lake monitoring

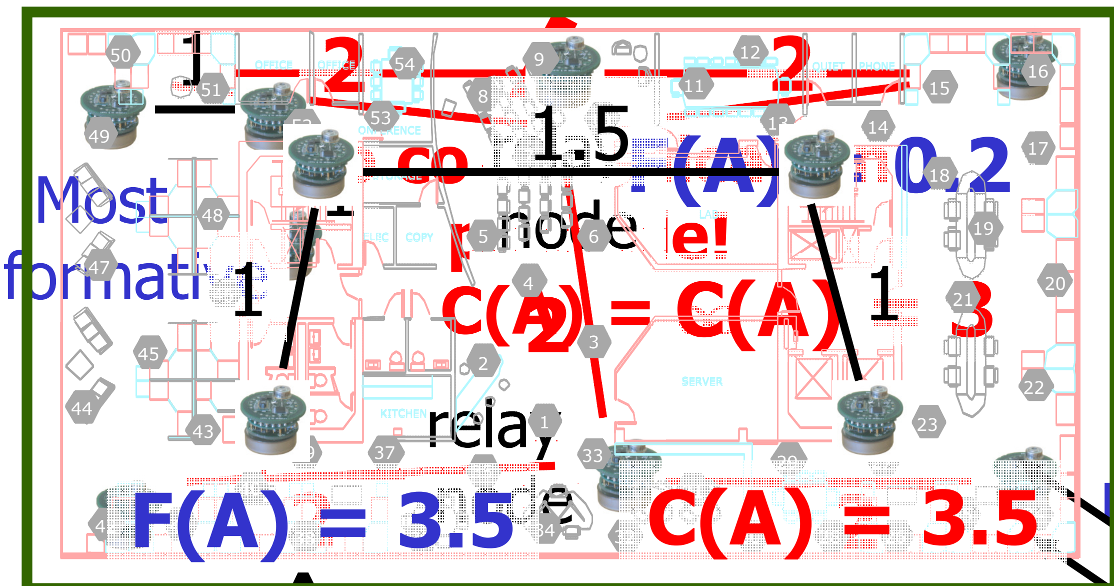
Sensors need to communicate (form a routing tree)



Building monitoring

Naïve approach: Greedy-connect

- Simple heuristic: **Greedily** optimize submodular utility function F(A)
- *Then* **add** nodes to minimize communication cost C(A)



F(A) = 4

C(A) = 10

Second most informative

efficient

Very informative,

High communication cost! ☹

Communication cost = Expected # of trials

(learned using Gaussian Processes)

Want to find optimal tradeoff between information and communication cost

The **pSPIEL** Algorithm
[Krause, Guestrin, Gupta, Kleinberg IPSN 2006]

- ***pSPIEL***: Efficient **nonmyopic** algorithm

  (**p**added **S**ensor **P**lacements at **I**nformative and cost-**E**ffective **L**ocations)

- **Decompose** sensing region into small, well-separated clusters
- Solve cardinality constrained problem **per cluster** (greedy)
- **Combine** solutions using k-MST algorithm

Guarantees for *pSPIEL*

[Krause, Guestrin, Gupta, Kleinberg IPSN 2006]

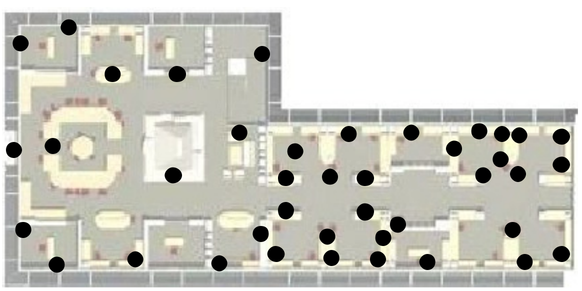
Theorem:

pSPIEL finds a tree T with

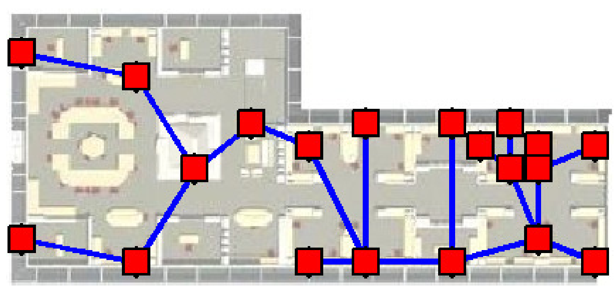
submodular utility	$F(T) \geq$	$\Omega(1)\ OPT_F$
communication cost	$C(T) \leq$	$O(\log \|V\|)\ OPT_C$

Proof of concept study

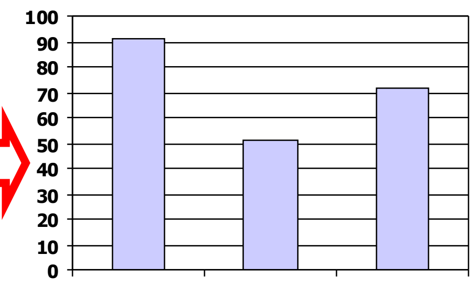
- Learned model from short deployment of 46 sensors at the Intelligent Workplace
- Manually selected 20 sensors;
  Used ***pSPIEL*** to place 12 and 19 sensors
- Compared prediction accuracy



Initial deployment and validation set

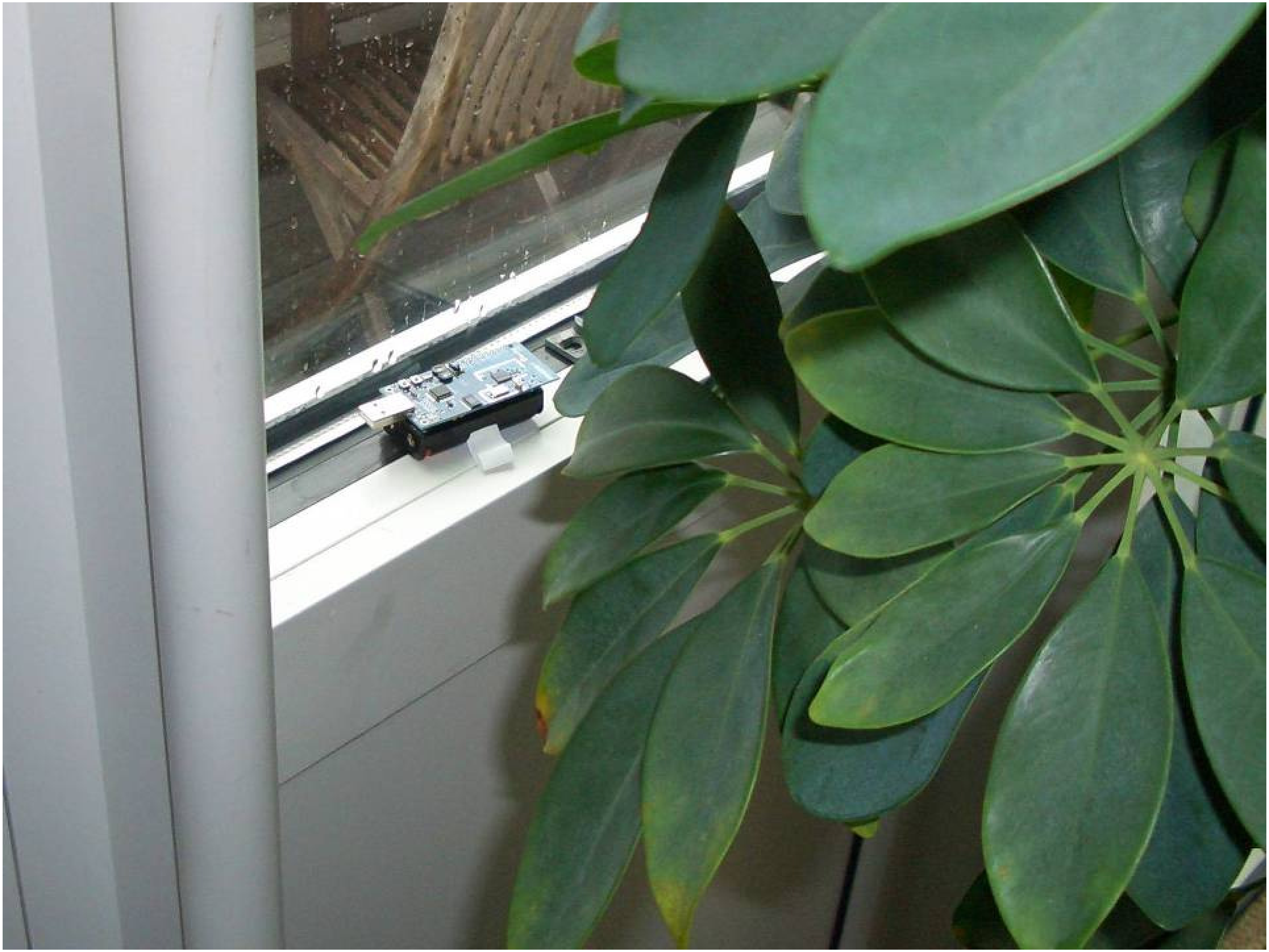


Optimized placements



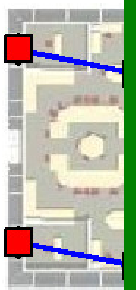
Accuracy

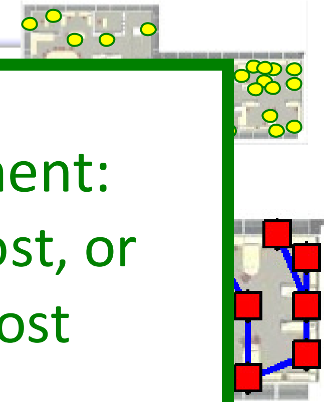
Time



Proof of concept study

accuracy on
46 locations





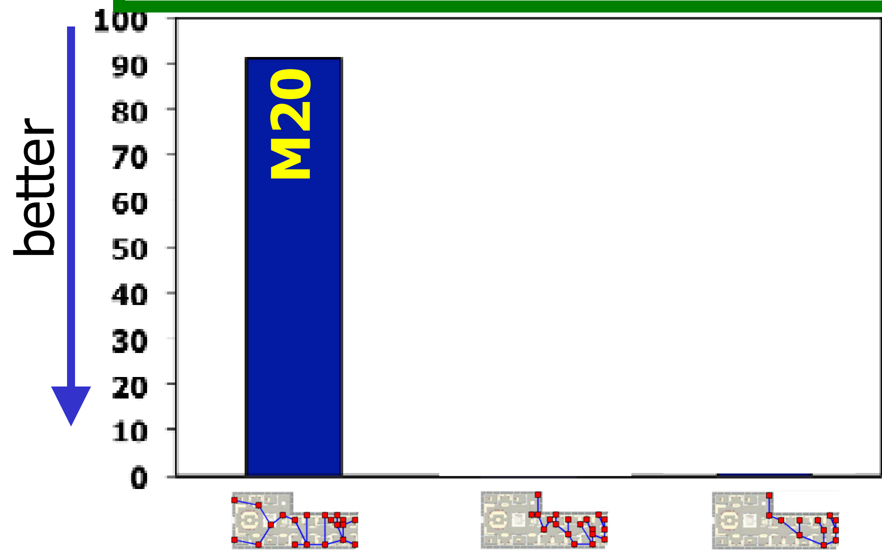
pSPIEL improves solution over intuitive manual placement:

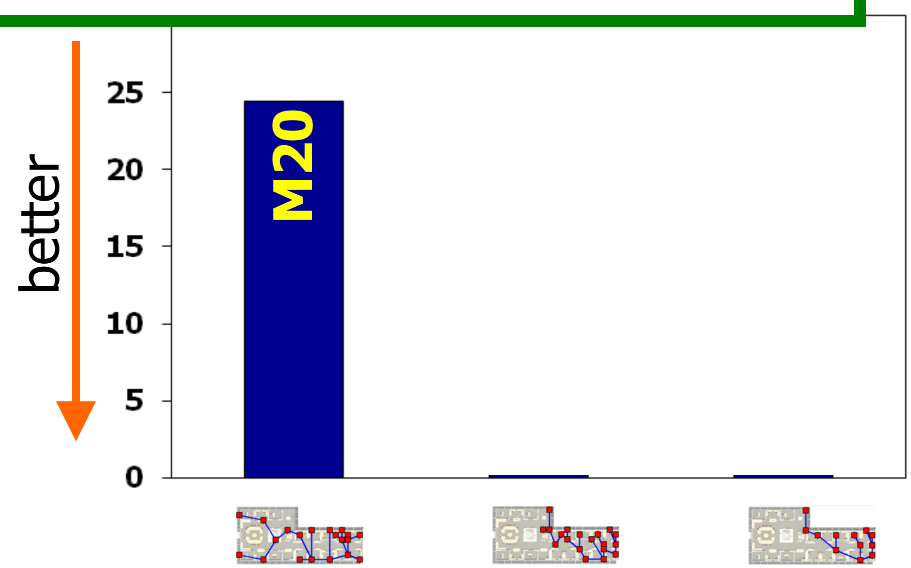
- 50% better prediction and 20% less communication cost, or
- 20% better prediction and 40% less communication cost

Poor placements can hurt a lot!

- Good solution can be unintuitive

Ro

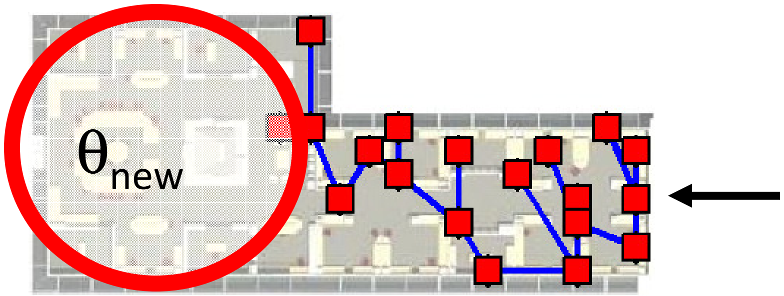




Robustness sensor placement

[Krause, McMahan, Guestrin, Gupta '07]

what if the usage pattern changes?

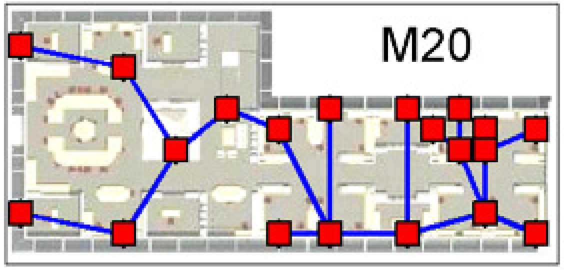


Optimal for old parameters θ_{old}

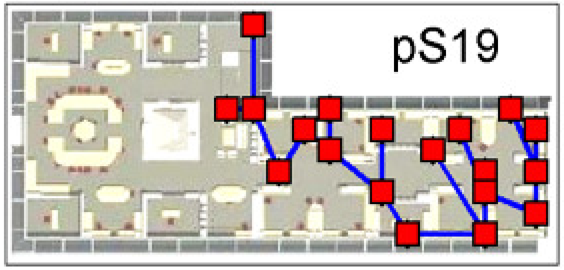
- Want placement to do well both under all possible parameters θ
  ➔ Maximize $\min_\theta F_\theta(A)$
- Unified view
  - Robustness to change in parameters
  - Robust experimental design
  - Robustness to adversaries

Can use *SATURATE* for robust sensor placement!

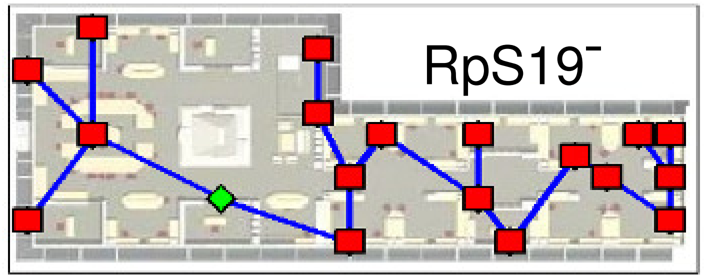
Robust pSpiel



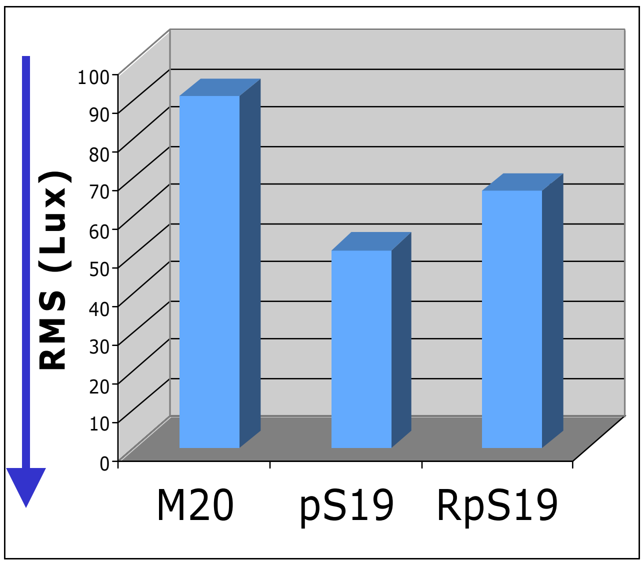
manual

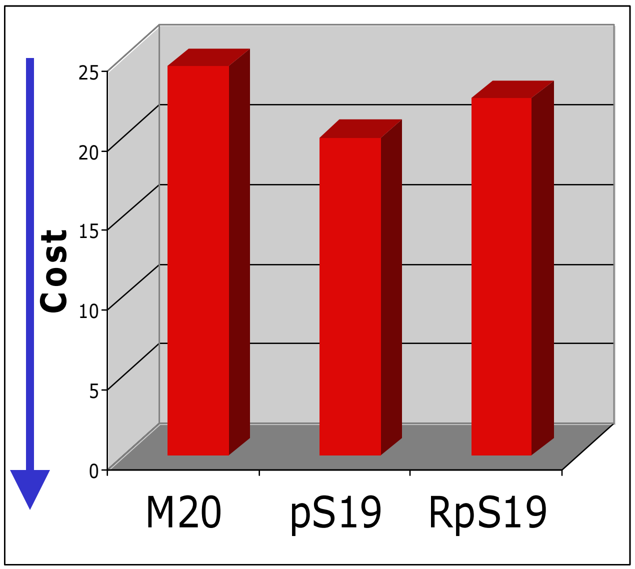


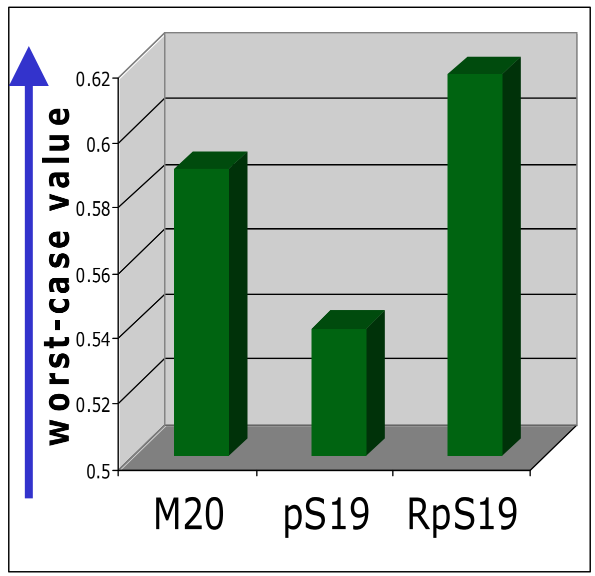
pSpiel



Robust pSpiel







Robust placement more intuitive, still better than manual!

Tutorial Overview

- Examples and properties of submodular functions ✓
  - Many problems submodular (mutual information, influence, …)
  - SFs closed under positive linear combinations; not under min, max
- Submodularity and convexity ✓
  - Every SF induces a convex function with SAME minimum
  - Special properties: Greedy solves LP over exponential polytope
- Minimizing submodular functions ✓
  - Minimization possible in polynomial time (but $O(n^8)$…)
  - Queyranne's algorithm minimizes symmetric SFs in $O(n^3)$
  - Useful for clustering, MAP inference, structure learning, …
- Maximizing submodular functions ✓
  - Greedy algorithm finds near-optimal set of k elements
  - For more complex problems (robustness, constraints) greedy fails, but there still exist good algorithms (*SATURATE*, *pSPIEL*, …)
  - Can get online bounds, lazy evaluations, …
  - Useful for feature selection, active learning, sensor placement, …
- Extensions and research directions

skip

Extensions and research directions

Learning submodular functions [Goemans, Harvey, Kleinberg, Mirrokni, '08]

- Pick m sets, $A_1 \ldots A_m$, get to see $F(A_1), \ldots, F(A_m)$
- From this, want to approximate F by F' s.t.

$$\boxed{1/\alpha \leq F(A)/F'(A) \leq \alpha \qquad \text{for all } A}$$

Theorem: Even if

- F is monotonic
- we can pick polynomially many A_i, chosen adaptively,

cannot approximate better than

$$\alpha = n^{1/2} / \log(n)$$

unless P = NP

Sequential selection
[Krause, Guestrin '07]

Thus far assumed know submodular function F (model of environment)

$\rightarrow$ **Bad assumption**

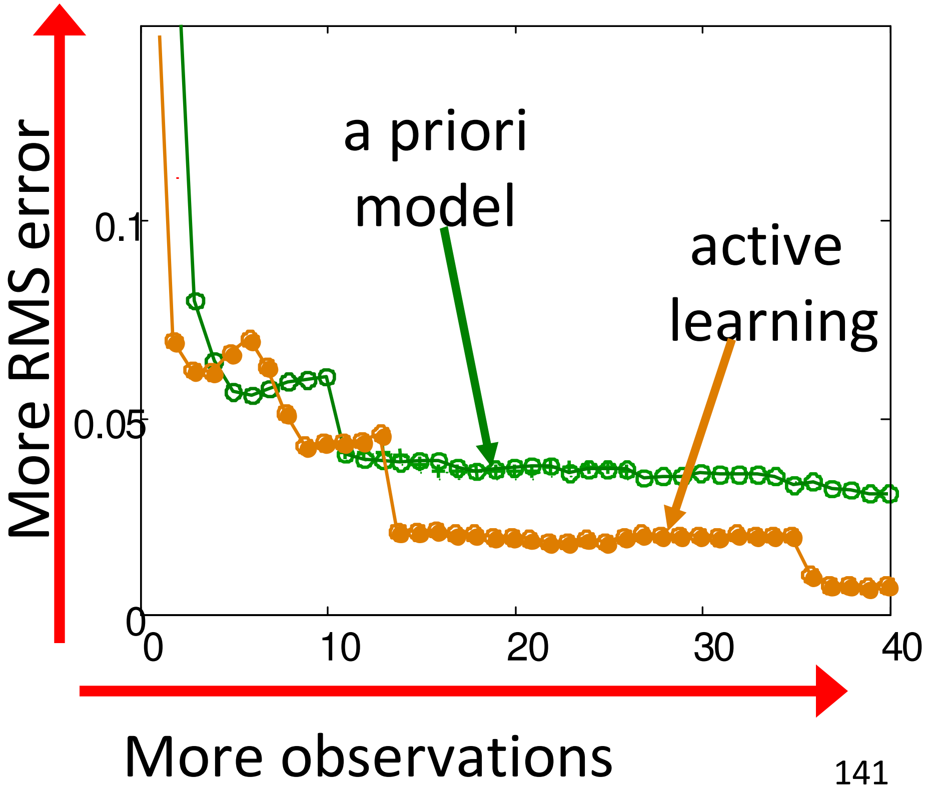
- Don't know lake correlations before we go…

Active learning:

Simultaneous sensing (selection) and model (F) learning

- Can use submodularity to analyze exploration/exploitation tradeoff
- Obtain theoretical guarantees

pH data from Merced river



Online maximization of submodular functions [Golovin & Streeter '07]

Pick sets	A_1	A_2	A_3	…	A_T	
SFs	F_1	F_2	F_3	…	F_T	
Reward	$r_1=F_1(A_1)$	r_2	r_3	…	r_T	Total: $\sum_t r_t \rightarrow \max$

Time

Theorem

Can efficiently choose $A_1, \ldots A_t$ s.t. in expectation

$$(1/T) \sum_t F_t(A_t) \geq (1/T)\,(1-1/e) \max_{|A| \leq k} \sum_t F_t(A)$$

for any sequence F_i, as $T \rightarrow \infty$

"Can asymptotically get 'no-regret' over clairvoyant greedy"

Game theoretic applications

How can we fairly distribute a set V of "unsplittable" goods to m people?

"Social welfare" problem:

- Each person i has submodular utility $F_i(A)$
- Want to partitition $V = A_1 \cup \ldots \cup A_m$ to maximize

$$\boxed{F(A_1,\ldots,A_m) = \sum_i F_i(A_i)}$$

Theorem [Vondrak, STOC '08]:
Can get 1-1/e approximation!

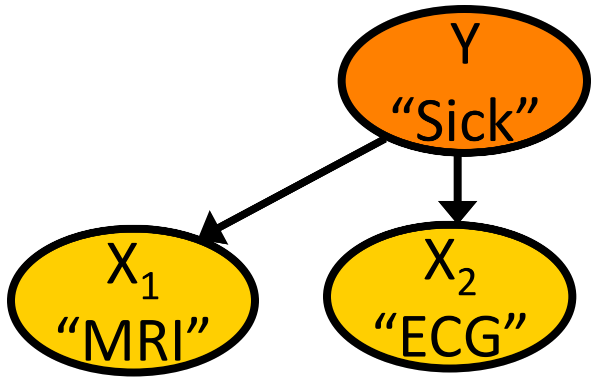
Beyond Submodularity: Other notions

- Posimodularity?
  - $F(A) + F(B) \geq F(A \setminus B) + F(B \setminus A) \;\; \forall A,B$
  - Strictly generalizes symmetric submodular functions
- Subadditive functions?
  - $F(A) + F(B) \geq F(A \cup B) \; \forall A,B$
  - Strictly generalizes monotonic submodular functions
- Crossing / intersecting submodularity?
  - $F(A) + F(B) \geq F(A \cup B) + F(A \cap B)$ holds for some sets A,B
  - Submodular functions can be defined on arbitrary lattices
- Bisubmodular functions?
  - Set functions defined on pairs (A,A') of disjoint sets of
  - $F(A,A') + F(B,B') \geq F((A,A') \vee (B,B')) + F((A,A') \wedge (B,B'))$
- Discrete-convex analysis (L-convexity, M-convexity, …)
- Submodular flows
- …

Beyond submodularity: Non-submodular functions

For F submodular and G supermodular, want

$$\boxed{A^* = \operatorname{argmin}_A F(A) + G(A)}$$



$F(\{X_1,X_2\}) \leq$
$F(\{X_1\})+F(\{X_2\})$

Example:

- –G (A) is information gain for feature selection
- F(A) is cost of computing features A, where "buying in bulk is cheaper"

In fact, any set function can be written this way!!

An analogy

- For F submodular and G supermodular, want

$$\boxed{A^* = \text{argmin}_A\ F(A) + G(A)}$$

- Have seen:

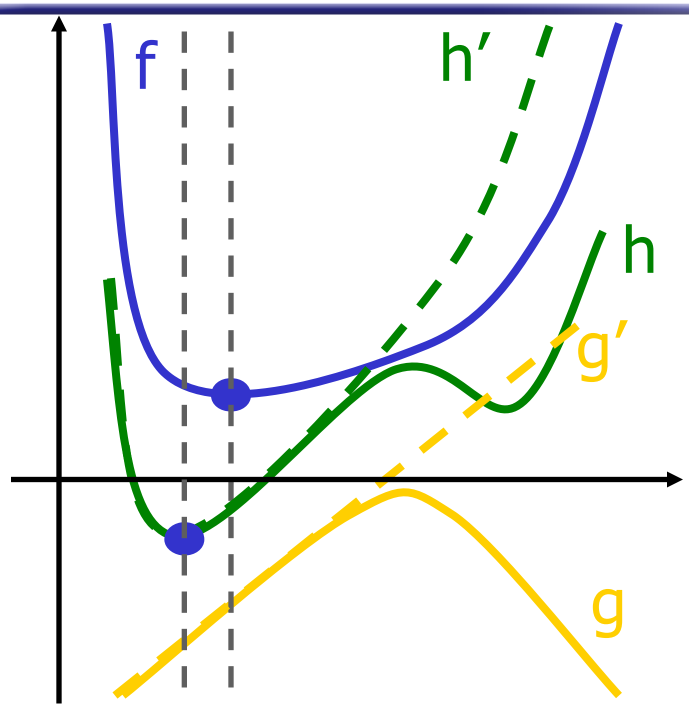
  submodularity ~ convexity

  supermodularity ~ concavity

- Corresponding problem: f convex, g concave

$$\boxed{x^* = \text{argmin}_x\ f(x) + g(x)}$$

DC Programming / Convex Concave Procedure [Pham Dinh Tao '85]



$x' \leftarrow$ argmin f(x)

While not converged do

1.) $g' \leftarrow$ linear upper bound of g, tight at x'

2.) $x' \leftarrow$ argmin f(x)+g'(x)

Will converge to local optimum
Generalizes EM, …

Clever idea [Narasimhan&Bilmes '05]:
Also works for submodular and supermodular functions!

- Replace 1) by "modular" upper bound
- Replace 2) by submodular function minimization

Useful e.g. for discriminative structure learning!

Many more details in their UAI '05 paper

Structure in ML / AI problems

ML last 10 years:

Convexity

Kernel machines
SVMs, GPs, MLE…

ML "next 10 years:"

Submodularity ☺

New structural properties

Structural insights help us solve challenging problems

Open problems / directions

Submodular optimization

- Improve on $O(n^8 \log^2 n)$ algorithm for minimization?
- Algorithms for constrained minimization of SFs?
- Extend results to more general notions (subadditive, …)?

Applications to AI/ML

- Fast / near-optimal inference?
- Active Learning
- Structured prediction?
- Understanding generalization?
- Ranking?
- Utility / Privacy?

Lots of interesting open problems!!

www.submodularity.org

- Examples and properties of submodular functions ✓
  - Many problems submodular (mutual information, influence, …)
  - SFs closed under positive linear combinations; not under min, max
- Submodularity and convexity ✓
  - Every SF induces a convex function with SAME minimum
  - Special properties: Greedy solves LP over exponential polytope
- Minimizing submodular functions ✓
  - Minimization possible in polynomial time (but $O(n^8)$…)
  - Queyranne's algorithm minimizes symmetric SFs in $O(n^3)$
  - Useful for clustering, MAP inference, structure learning, …
- Maximizing submodular functions ✓
  - Greedy algorithm finds near-optimal set of k elements
  - For more complex problems (robustness, constraints) gre… exist good algorithms (*SATURATE*, *pSPIEL*, …)
  - Can get online bounds, lazy evaluations, …
  - Useful for feature selection, active learning, sensor place…
- Extensions and research directions ✓
  - Sequential, online algorithms
  - Optimizing non-submodular functions

Check out our Matlab toolbox!

sfo_queyranne,
sfo_min_norm_point,
sfo_celf, sfo_sssp,
sfo_greedy_splitting,
sfo_greedy_lazy,
sfo_saturate,
sfo_max_dca_lazy
…