Combinatorial optimization and graphical models

Submodular functions

Optimization - 10725
Carlos Guestrin
Carnegie Mellon University

April 28th, 2008

Most probable explanation (MPE) in a Markov network

- Markov net:
- Most probable explanation:
- In general, NP-complete problem, and hard to approximate

MPE for attractive MNs – 2 classes

- Attractive MN:
  - E.g., image classification
- Finding most probable explanation
- Can be solved by

MPE, Attractive MNs, k classes

- MPE for k classes:
- Multiway cut:
  - Graph G, edge weights w_{ij}
  - Finding minimum cut, separate $s_1,\ldots,s_k$
- Multiway cut problem is

Multiway cut – combinatorial algorithm

- Very simple alg:
  - For each i=1…k
    - Find cut C_i that separates s_i from rest
  - Discard $\text{argmax}_i\ w(C_i)$, return union of rest
- Algorithm achieves 2-2/k approximation
  - OPT cut A* separates graph into k components
    - No advantage in more than k
  - From A* form $A^*_1,\ldots,A^*_k$, where A^*_i separates s_i from rest
  - Each edge in A* appears in
    - Thus

Multiway cut proof

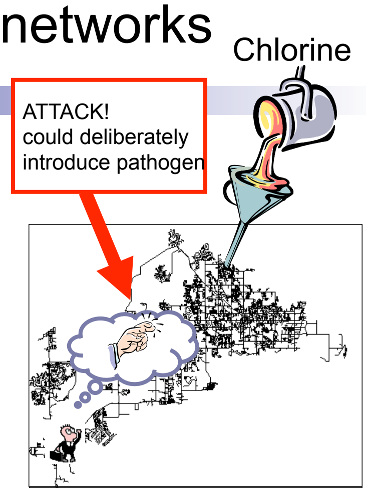
- Thus, for OPT cut A* we have that:
- Each A^*_i separates s_i from rest, thus
- But, can do better, because

General solutions to combinatorial problems

- Nonlinear optimization
  - Extremely general
  - NP-Hard to solve
- Covex problems:
  - Special problem structure
  - General efficient solution algorithms
  - Applies to many continuous problems
- Combinatorial optimization
  - Integer programming
    - Very general, but NP-hard
  - Convex relaxations
    - Approximation guarantees for many problems
    - Usually solution is problem specific
      - Though there are general principles
- Is there general problem structure in combinatorial problems?
  - Analogously to convexity?
    - Recognize structure and use general algorithms?

Water distribution networks

- Water distribution in a city → very complex system
- Pathogens in water can affect thousands (or millions) of people
- Currently: Add chlorine to the source and hope for the best

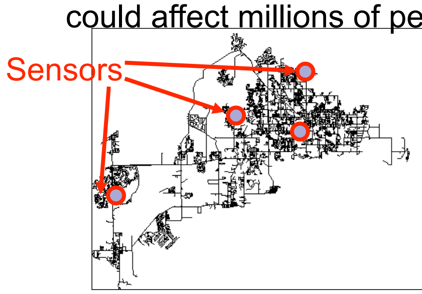


Simulator from EPA

Monitoring water networks

[Krause, Leskovec, G., Faloutsos, VanBriesen '08]

- Contamination of drinking water could affect millions of people



Simulator from EPA

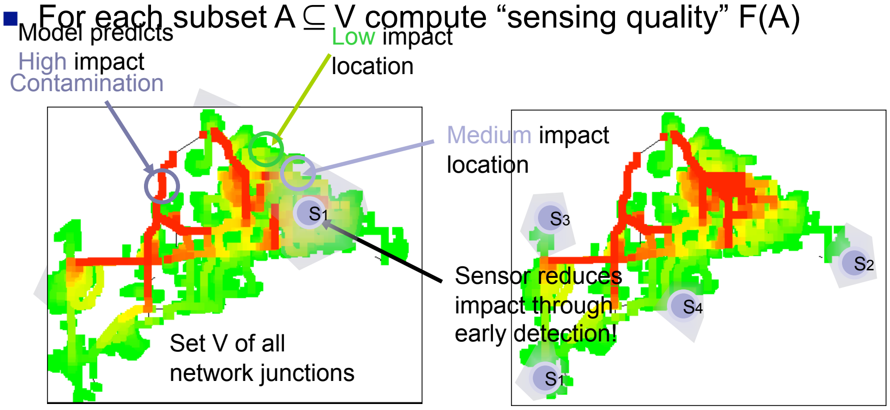
Hach Sensor
~$14K

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

Where should we place sensors to detect contaminations quickly ?

Model-based sensing

- Model predicts impact of contaminations
  - For water networks: Water flow simulator from EPA
  - For lake monitoring: Learn probabilistic models from data (later)
- For each subset A ⊆ V compute "sensing quality" F(A)



High sensing quality F(A) = 0.9

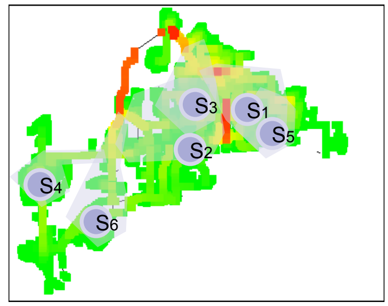
Low sensing quality F(A)=0.01

Sensor placement

- Given: finite set V of locations, sensing quality F
- Want: $A^* \subseteq V$ such that

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

Typically NP-hard!

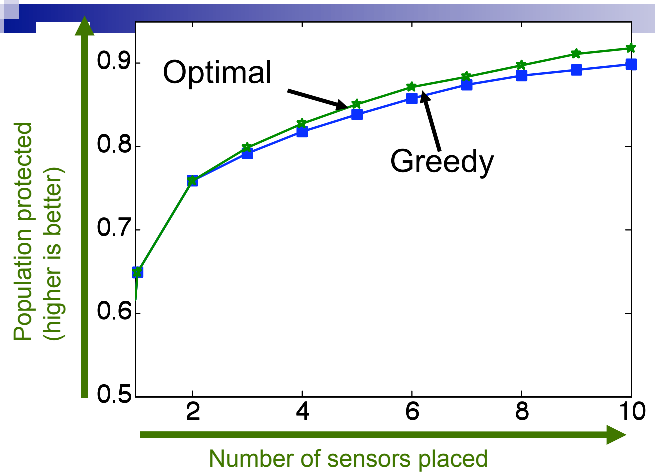


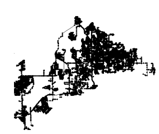
<u>Greedy algorithm:</u>

Start with A = Ø ;
For i = 1 to k
  $s^* := \operatorname{argmax}_s F(A \cup \{s\})$
  $A := A \cup \{s^*\}$

How well can this simple heuristic do?

Performance of greedy algorithm



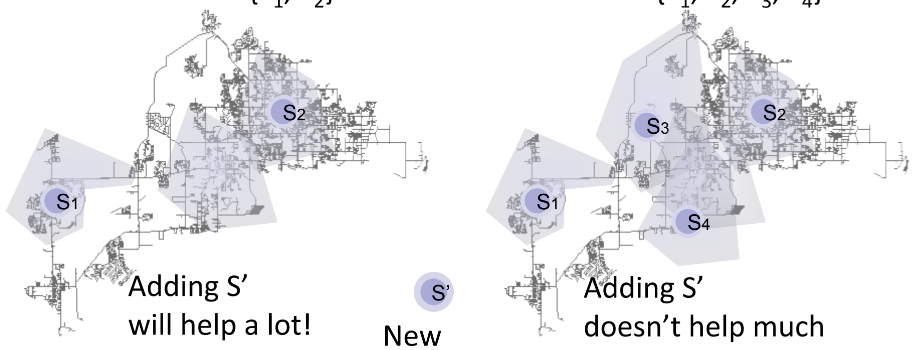


Small subset of Water networks data

Greedy score empirically close to optimal. Why?

Key property: Diminishing returns

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



New
sensor S'

Theorem [Krause, Leskovec, G., Faloutsos, VanBriesen '08]:

Sensing quality F(A) in water networks is submodular!

Submodularity

- Formalizes notion of diminishing returns

- Equivalent definition:

What do we get from submodularity

- Submodularity is a general property of set functions
- Submodular function can be minimized in polynomial time!
- But our problem is

Another example…

- Maximum cover
  - Ground elements
  - Set of sets
  - Pick k sets, maximize number of covered elements

Maximizing submodular functions – cardinality constraint

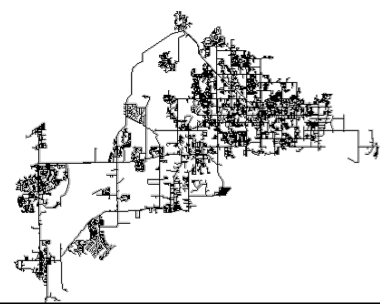
- Given
  - Submodular function
  - Normalized
  - Non-decreasing
- Greedy algorithm guarantees
- Can you get better algorithm?

Online bounds

- Submodularity provides bounds on the quality of the solution A obtained by any algorithm
  - For normalized, non-decreasing functions
- Advantage of adding elements to A:
- Bound on the quality of any set A:
- Tighter bound:

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"

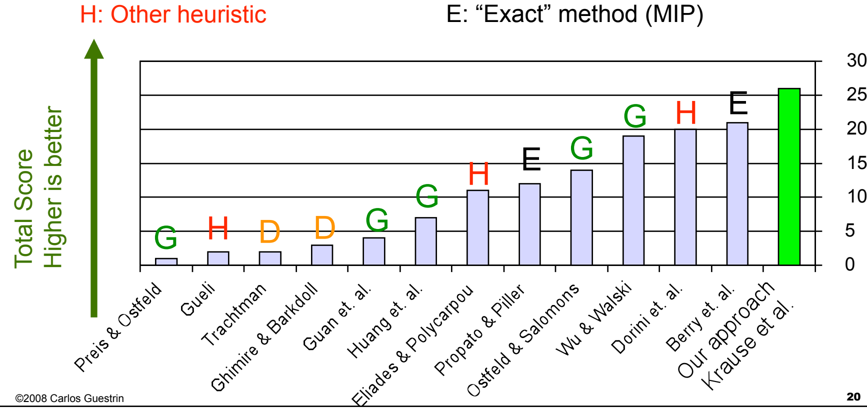


BWSN Competition results

- 13 participants
- Performance measured in 30 different criteria

G: Genetic algorithm D: Domain knowledge

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

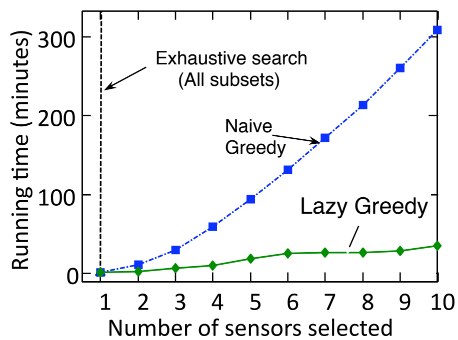


What was the trick?

Simulated all 3.6M contaminations on 2 weeks / 40 processors

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

➔ Very accurate sensing quality ☺ 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! ☺

Lazy evaluations

- Naïve implementation of greedy:
- Advantage of an element never increases:
  - Advantage:
  - What if you already picked a larger set:
    - Set after picking i elements: A_i
- Lazy evaluations:
  - Keep a priority queue over elements:
    - Initialize with advantage of each element
  - Pick element on top, recompute priority
  - If element remains on top

Other maximization settings

- Non-monotone submodular functions
- Non-unit costs
- Complex constraints
  - Paths
  - Spanning trees
- Worst-case optimization