Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Submodular Optimization in Computational Sustainability

Informatik
Computer Science

Andreas Krause

Master Class at CompSust 2012

Combinatorial optimization in computational sustainability

- Many applications in computational sustainability require solving large discrete optimization problems:

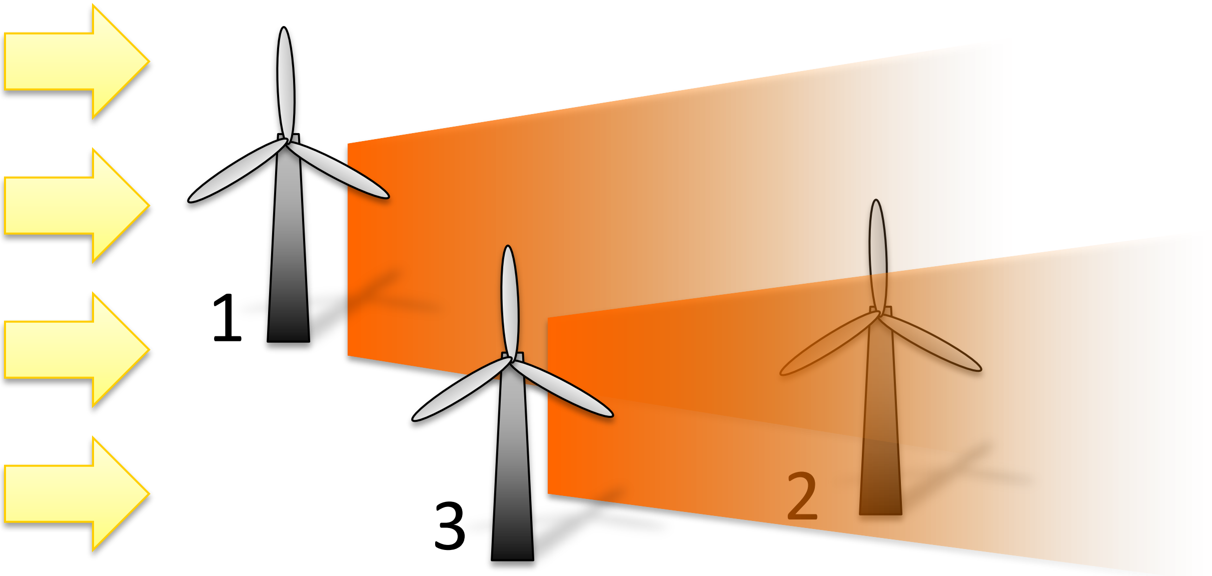
- Given finite set V wish to select subset A (subject to some constraints) maximizing utility F(A)

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

- These problems are the focus of this tutorial.

Wind farm Deployment

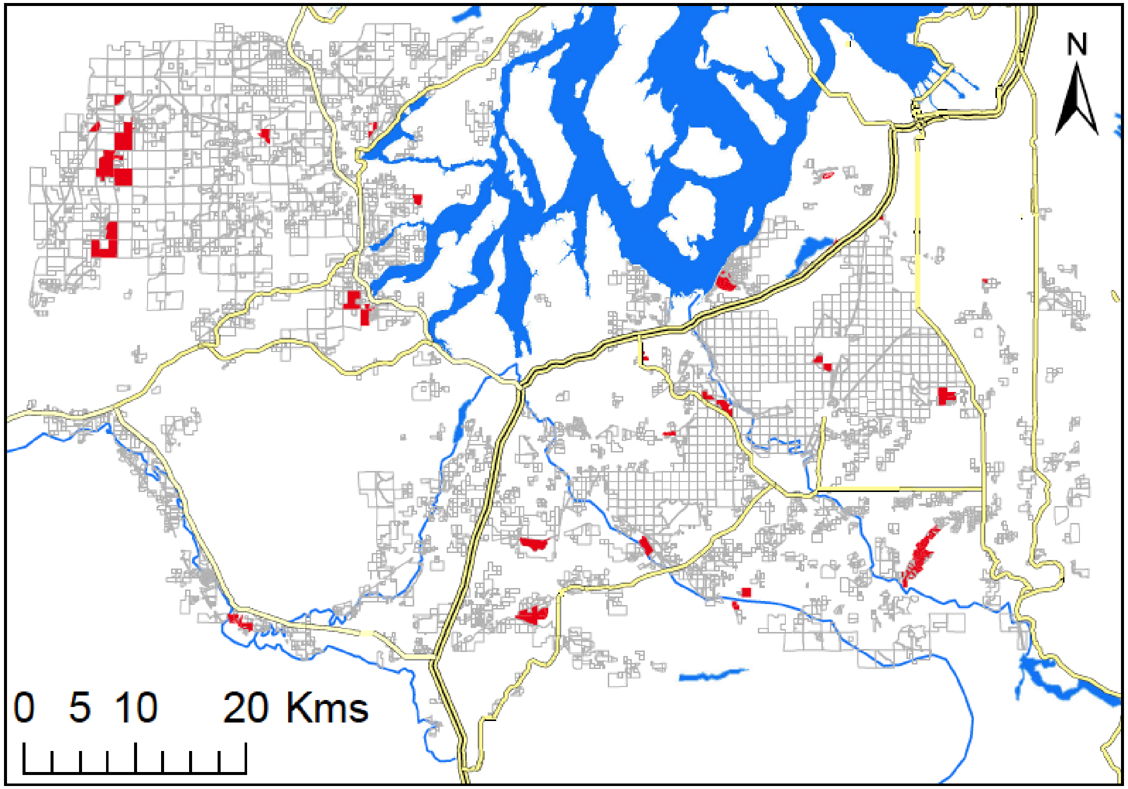
[Changshui et al, Renewable Energy, 2011]



How should we deploy wind farms to maximize efficiency?

Conservation Planning

[w Golovin, Converse, Gardner, Morey – AAAI '11 Outstanding Paper]



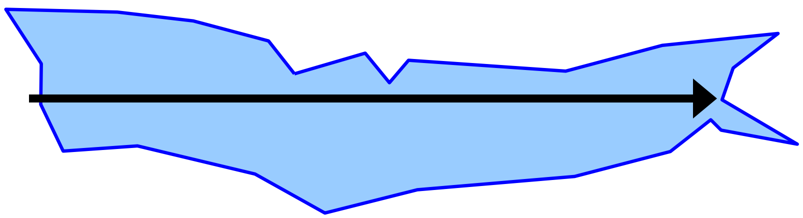
Which patches of land should we recommend?

Robotic monitoring of rivers and lakes

[with Singh, Guestrin, Kaiser, Journal of AI Research ’09]

Need to monitor large spatial phenomena

Temperature, nutrient distribution, fluorescence, …

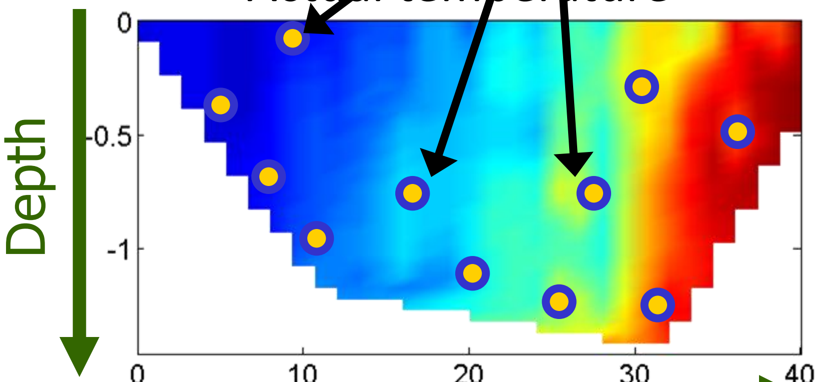


NIMS

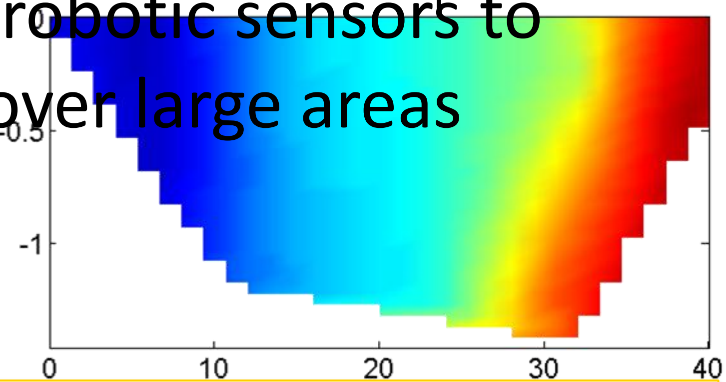
Kaiser et.al. (UCLA)

Can only make a limited number of measurements!

Actual temperature



Predicted temperature



Use robotic sensors to cover large areas

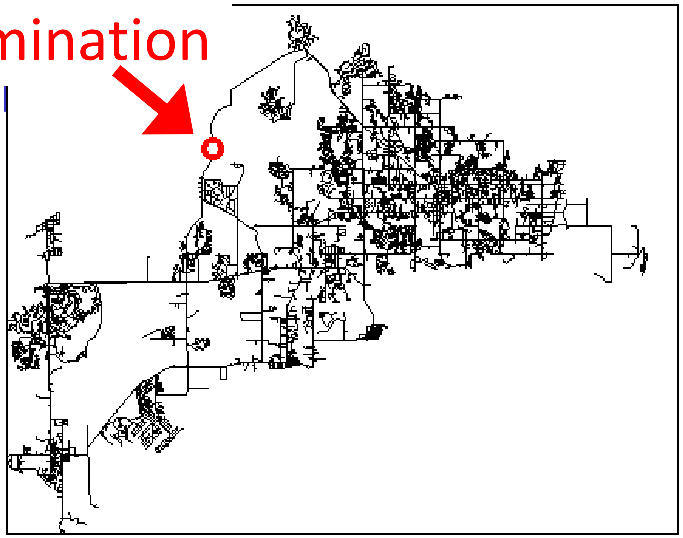
Where should we sense to get most accurate predictions?

Monitoring water networks

[with Leskovec, Guestrin, VanBriesen, Faloutsos, J Wat Res Mgt '08]

Contamination of drinking water could affect millions of people

Contamination



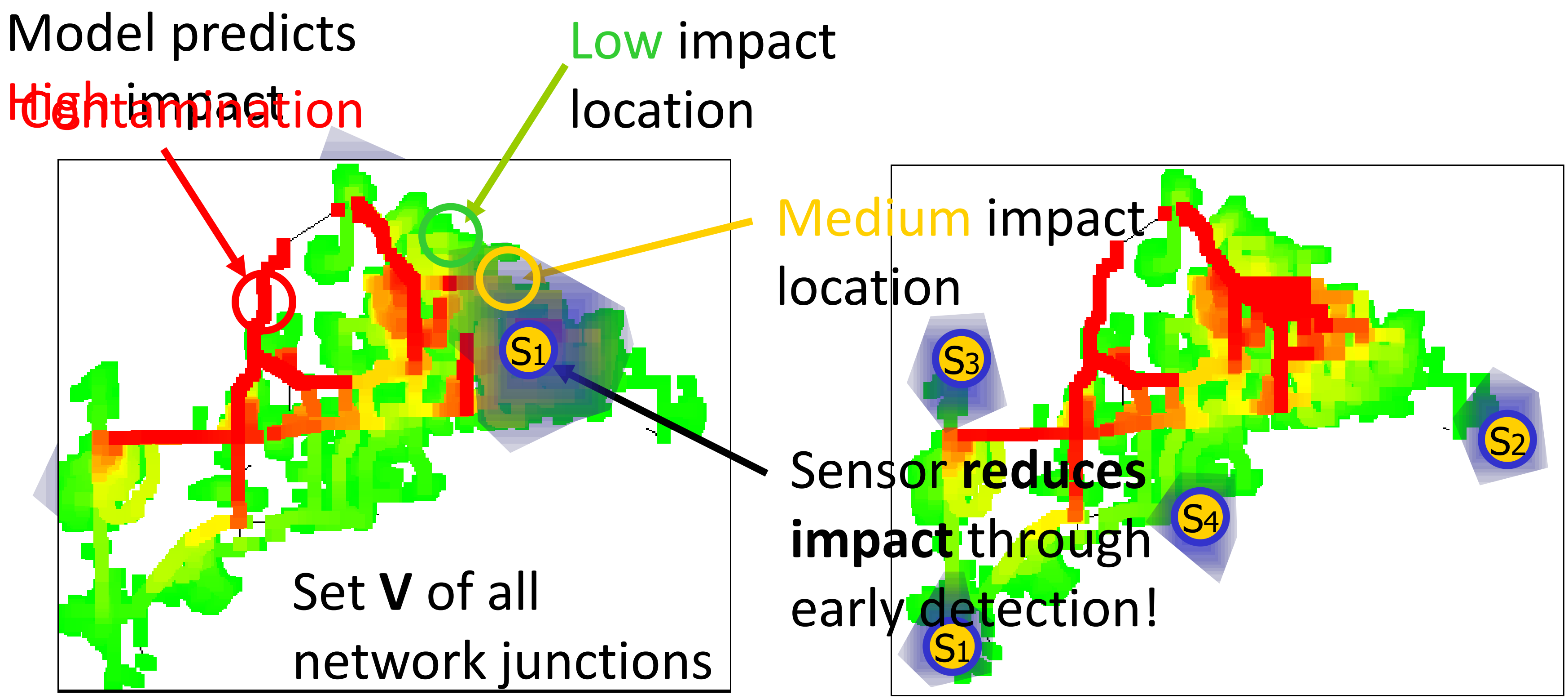
~$14K

Place sensors to detect contaminations

Where should we place sensors to quickly detect contamination?

Quantifying utility of sensor placements

- Model predicts impact of contaminations
- For each subset A of V compute *sensing quality* F(A)



High sensing quality F(A) = 0.9

Low sensing quality F(A)=0.01

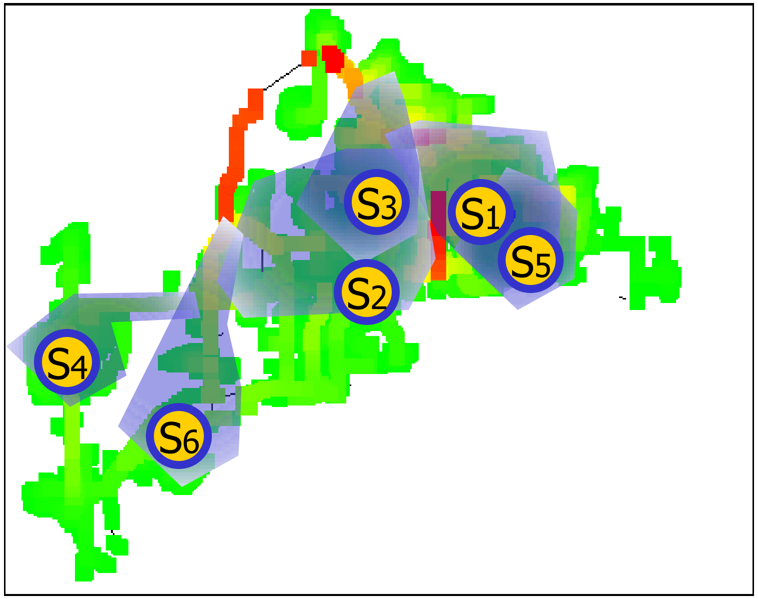
Sensor placement

Given: finite set V of locations, sensing quality F

Want: $\mathcal{A}^* \subseteq \mathcal{V}$ such that

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

NP-hard!



Greedy algorithm:

Start with A = {}

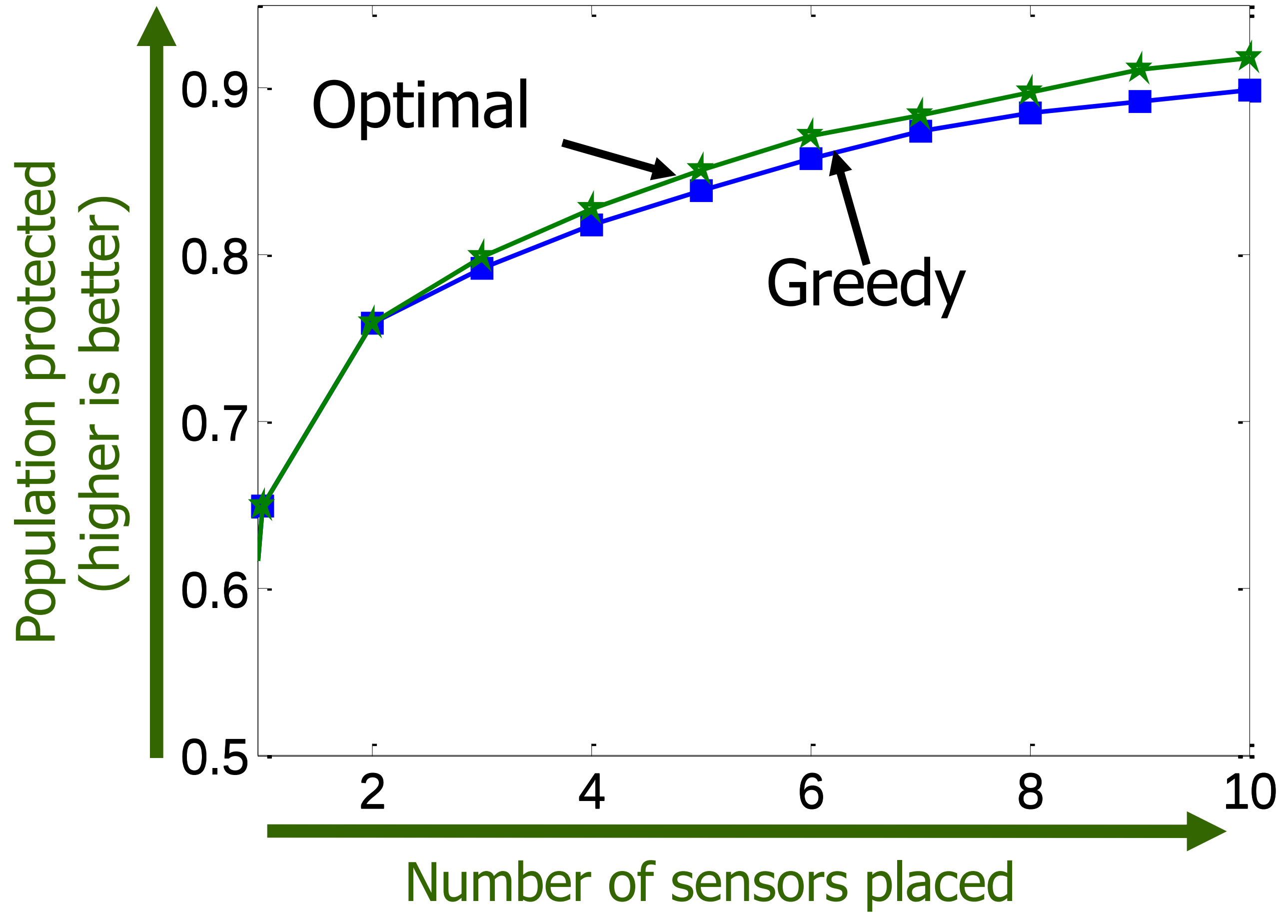
For i = 1 to k

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

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

How well can this simple heuristic do?

Performance of greedy algorithm

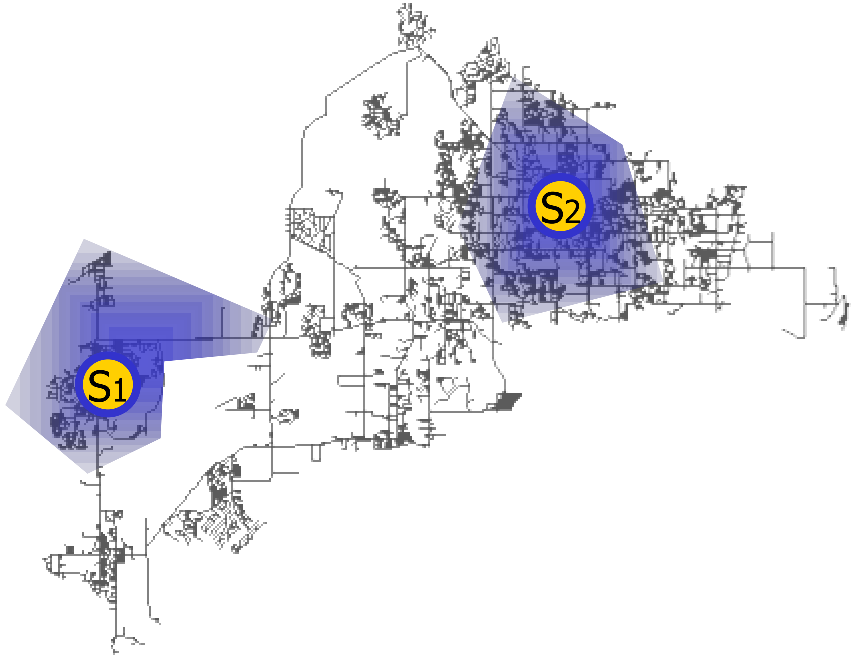


Small subset of Water networks data

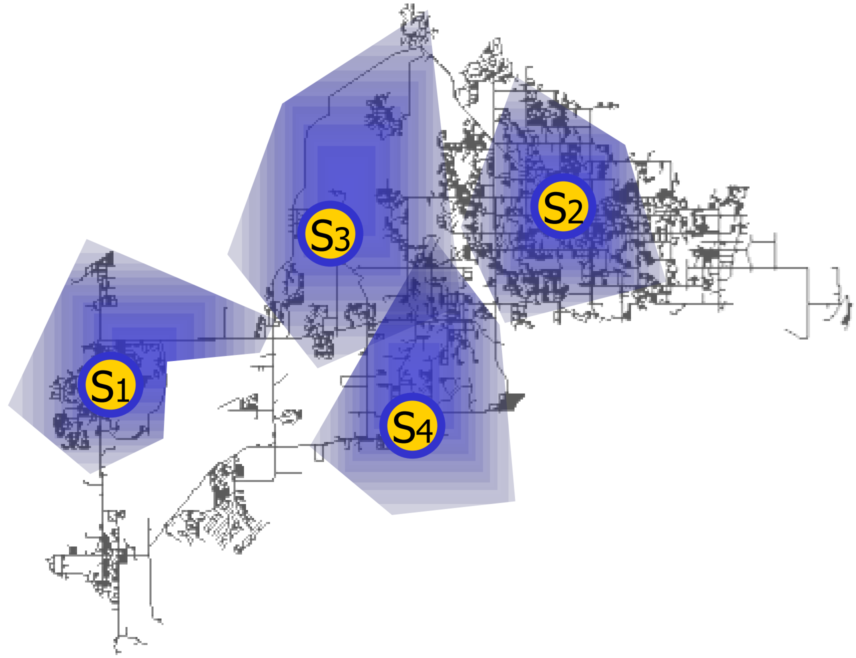
Greedy score empirically close to optimal. Why?

Key property 1: Monotonicity

Placement A = $\{S_1, S_2\}$



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

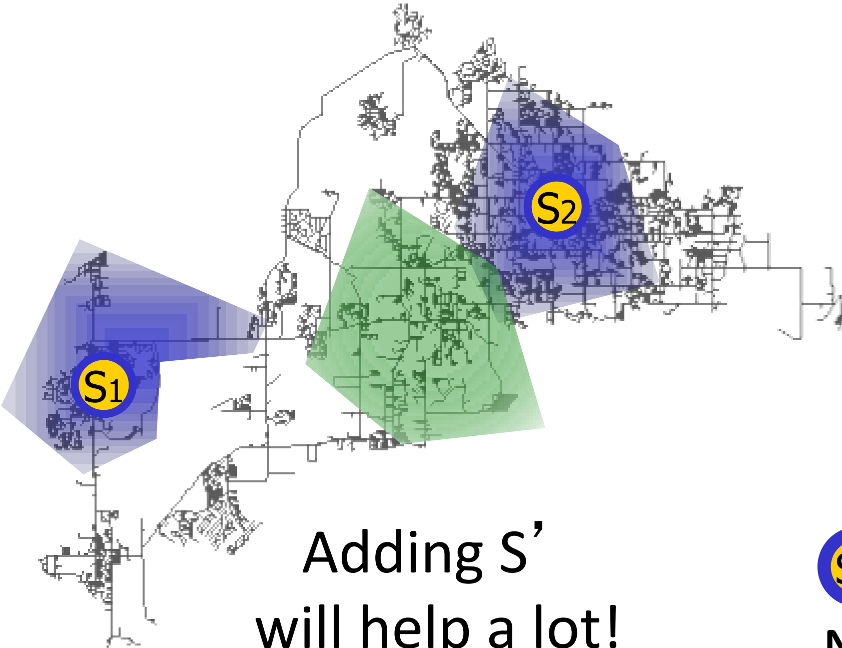


F is monotonic: $$\forall A \subseteq B : F(A) \leq F(B)$$

Adding sensors can only help

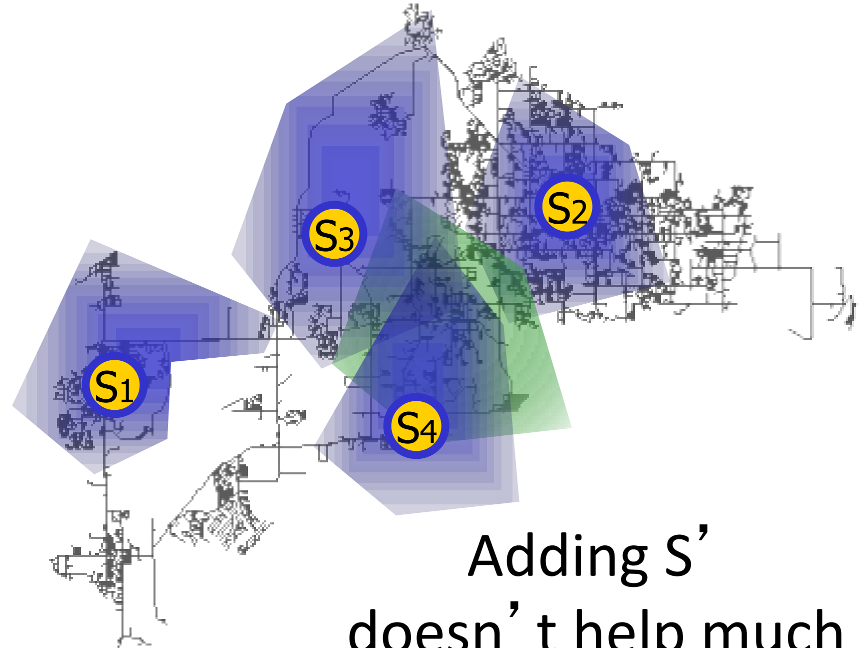
Key property 2: Diminishing returns

Placement A = {S_1, S_2}



Adding S'
will help a lot!

Placement B = {S_1, S_2, S_3, S_4}

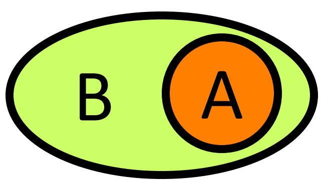


Adding S'
doesn't help much

New
sensor S'

New sensor Y'

Submodularity:



+• S' Large improvement

+• S' Small improvement

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

One reason submodularity is useful

Theorem [Nemhauser et al '78]

Suppose F is *monotonic* and *submodular*. Then greedy algorithm gives constant factor approximation:

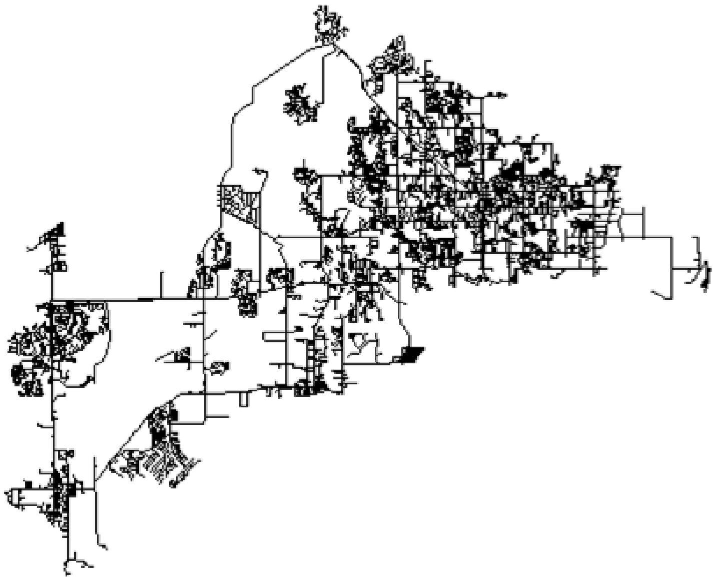
$$F(A_{greedy}) \geq \underbrace{(1 - 1/e)}_{\sim 63\%} \max_{|A| \leq k} F(A)$$

- Greedy algorithm gives near-optimal solution!
- In general, guarantees best possible unless P = NP!

Battle of the Water Sensor Networks Competition

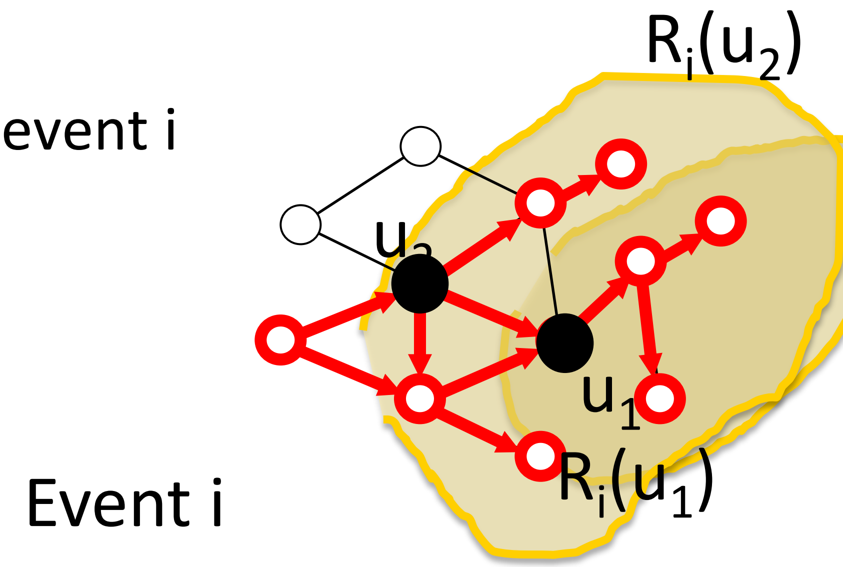
[with Leskovec, Guestrin, VanBriesen, Faloutsos, J Wat Res Mgt 2008]

- 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"



Reward function is submodular

- <u>Claim:</u>
  Reward function is monotonic submodular
- Consider event *i*:
  - $R_i(u_k)$ = benefit from sensor u_k in event i
  - $R_i(A) = \max R_i(u_k),\ u_k \in A$
  - $\Rightarrow R_i$ is submodular
- Overall objective:
  - $F(A) = \sum \text{Prob}(i)\, R_i(A)$
  - Submodular??



Closedness properties

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

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, $\lambda_i>0 \Rightarrow \sum_i \lambda_i F_i(A)$ submodular

Reward function is submodular

- <u>Claim:</u>
  Reward function is monotonic submodular

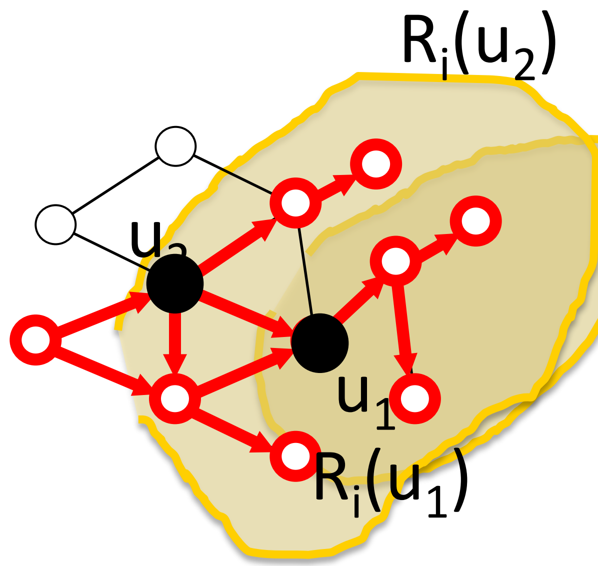
- Consider event *i*:
  - $R_i(u_k)$ = benefit from sensor u_k in event i
  - $R_i(A) = \max R_i(u_k),\ u_k \in A$
  - $\Rightarrow R_i$ is submodular

- Overall objective:
  - $F(A) = \sum \text{Prob}(i)\, R_i(A)$

➔ F is submodular!

➔ Can use greedy algorithm to solve $\max_{|A| \leq k} F(A)$!

Event i

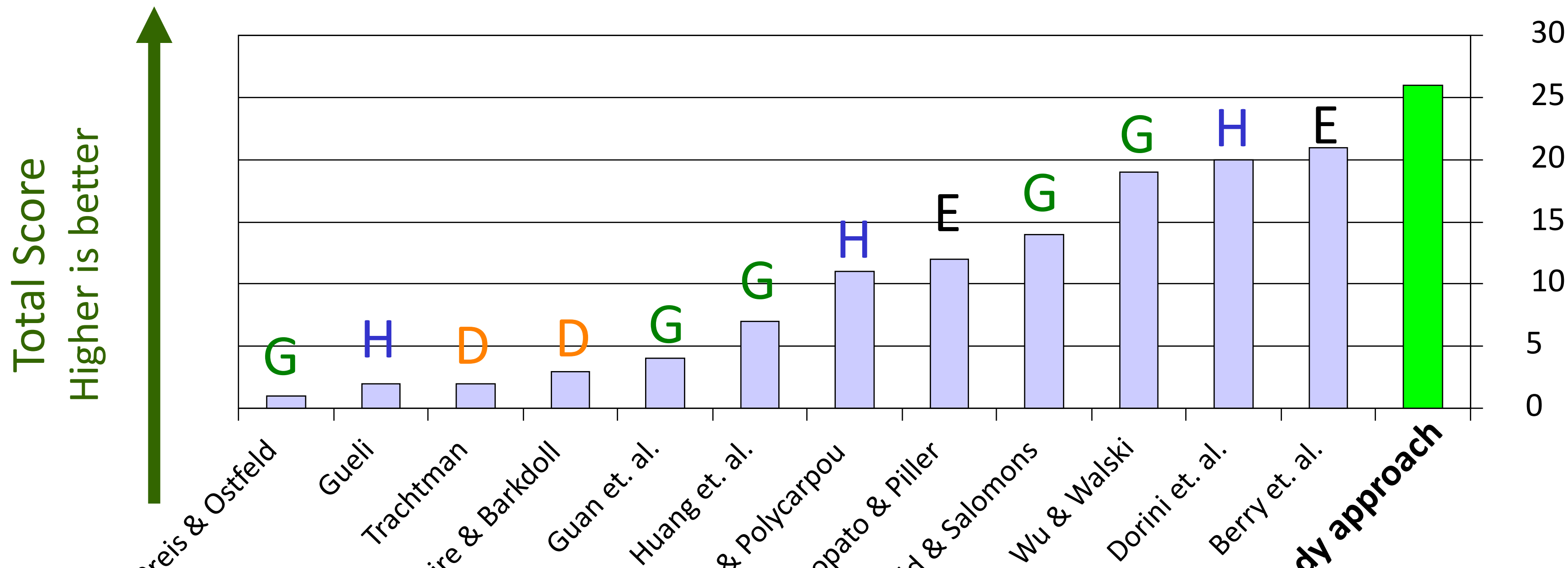


BWSN Competition results

- 13 participants
- Performance measured in 30 different criteria

G: Genetic algorithm D: Domain knowledge

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



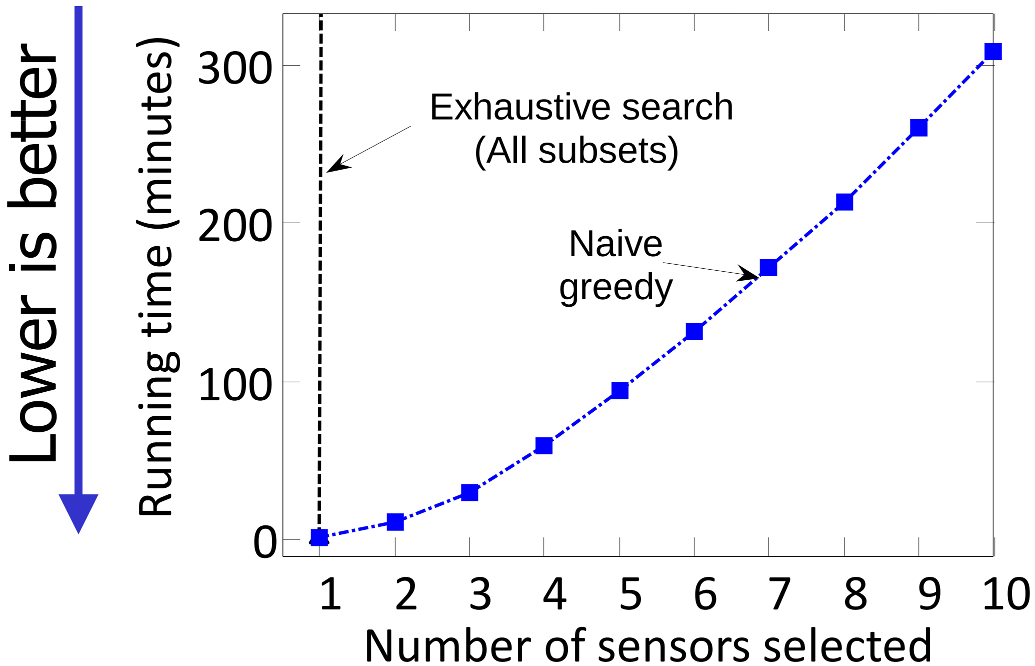
24% better performance than runner-up! ☺

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

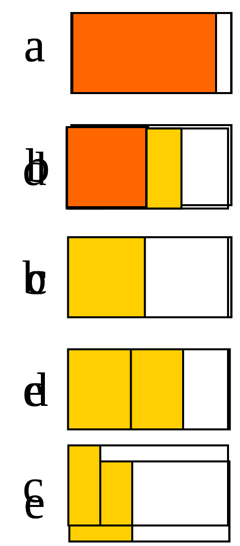
30 settings ☹

"Lazy" greedy algorithm [Minoux '78]

$$\Delta(s \mid A) = F(A \cup \{s\}) - F(A)$$

Lazy greedy algorithm:

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



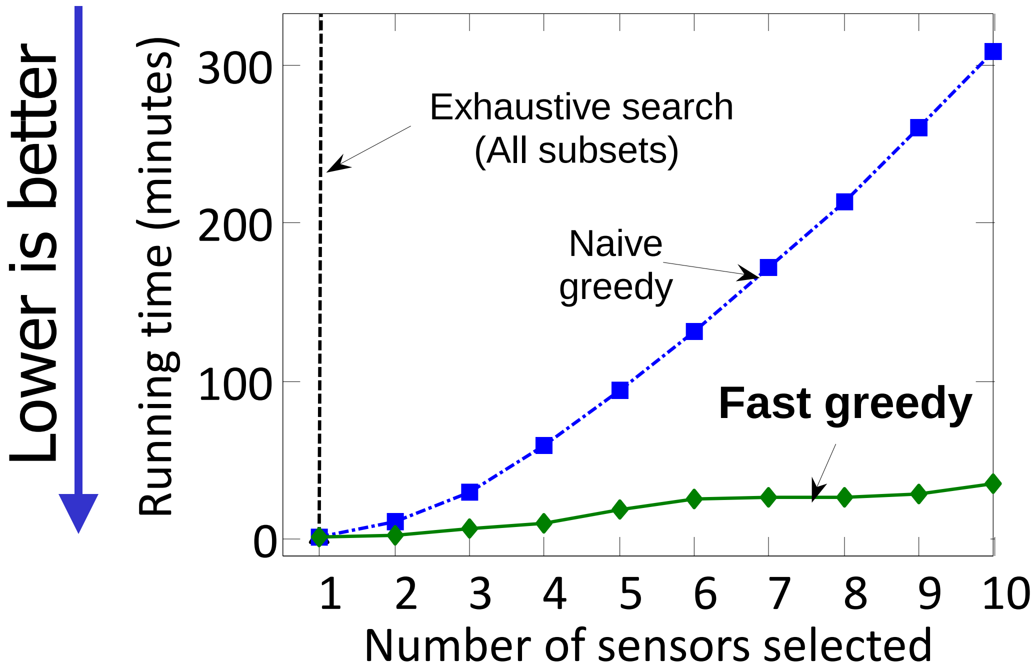
Note: Very easy to compute online bounds, use in other algo's, etc. [Leskovec, Krause et al. '07]

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 ☹

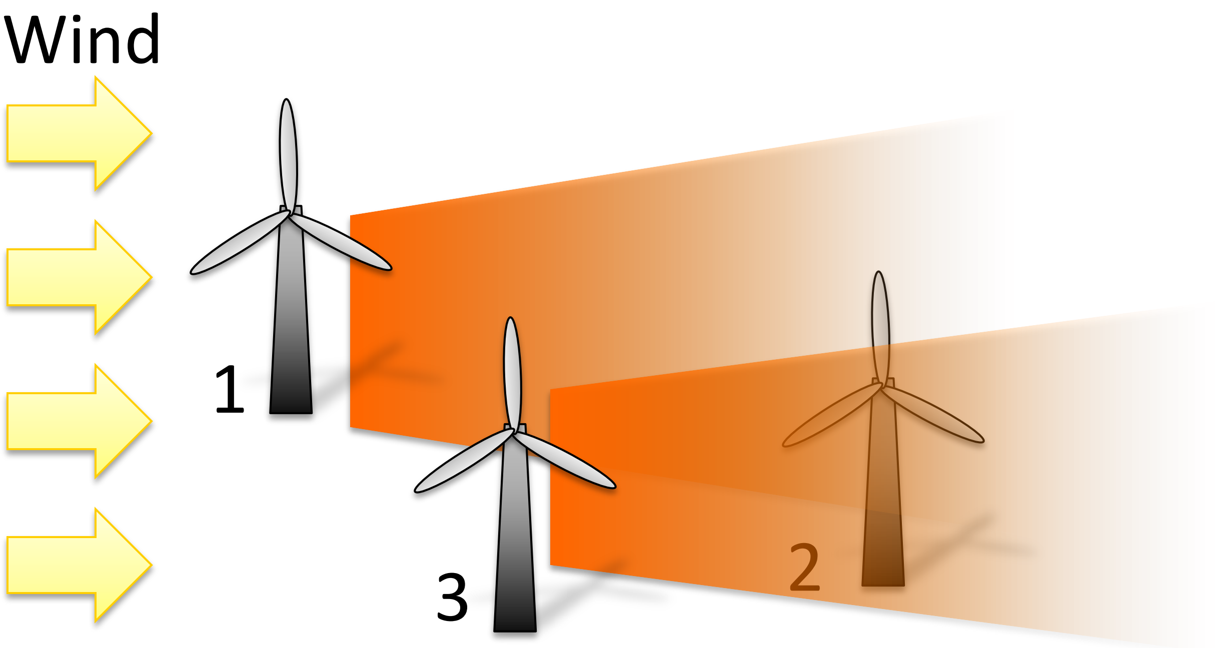
Submodularity to the rescue:

Using "lazy evaluations": 1 hour/20 sensors

Done after 2 days! ☺

Example: Windfarm deployment

[Changshui et al, Renewable Energy, 2011]



Contribution of 2
reduced due to wake effects

$$F(\{1,2\}) - F(\{1\}) \geq F(\{1,2,3\}) - F(\{1,3\})$$

Total power F(A) is monotonic submodular! ☺

Example: Windfarm deployment

[Changshui et al, Renewable Energy, 2011]

	Greedy	**LazyGreedy**	**Genetic Algo**
Power (kW)	79,585	79,585	78,850
Runtime	2.5 min	10 sec	1.6 hours

Other interesting directions

- Many sensing problems involve maximization of monotonic submodular functions
  - Can use greedy algorithm to get near-optimal solutions!
  - Lazy evaluations provide dramatic speedup

- How can we handle more complex settings:
  - Complex constraints / complex cost functions?
  - Sequential decisions?

Non-constant cost functions

- For each s □ V, let c(s)>0 be its cost (e.g., conservation cost, hardware cost, …)
- Cost of a set $C(A) = \sum_{s \in A} c(s)$
- Want to solve

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

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

Start with A := {};

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 U {s*}

Performance of cost-benefit greedy

Set A	F(A)	C(A)
{a}	2ε	ε
{b}	1	1

Want

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

Cost-benefit greedy picks a.

Then cannot afford b!

➔ Cost-benefit greedy performs arbitrarily badly!

Cost-benefit optimization
[Wolsey '82, Sviridenko '04, Krause et al '05]

Theorem [Krause and Guestrin '05]

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

Then

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

~39%

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]
- (1-1/e) approximation for multiple linear constraints [Kulik '09]
- 0.38/k approximation for k matroid and m linear constraints [Chekuri et al '11]

Application: Conservation Planning

[w Golovin, Converse, Gardner, Morey – AAAI '11 Outstanding Paper]

How should we select land for conservation to protect rare & endangered species?

Case Study: Planned Reserve in Washington State

Mazama pocket gopher

streaked horned lark

Taylor's checkerspot

Problem Ingredients

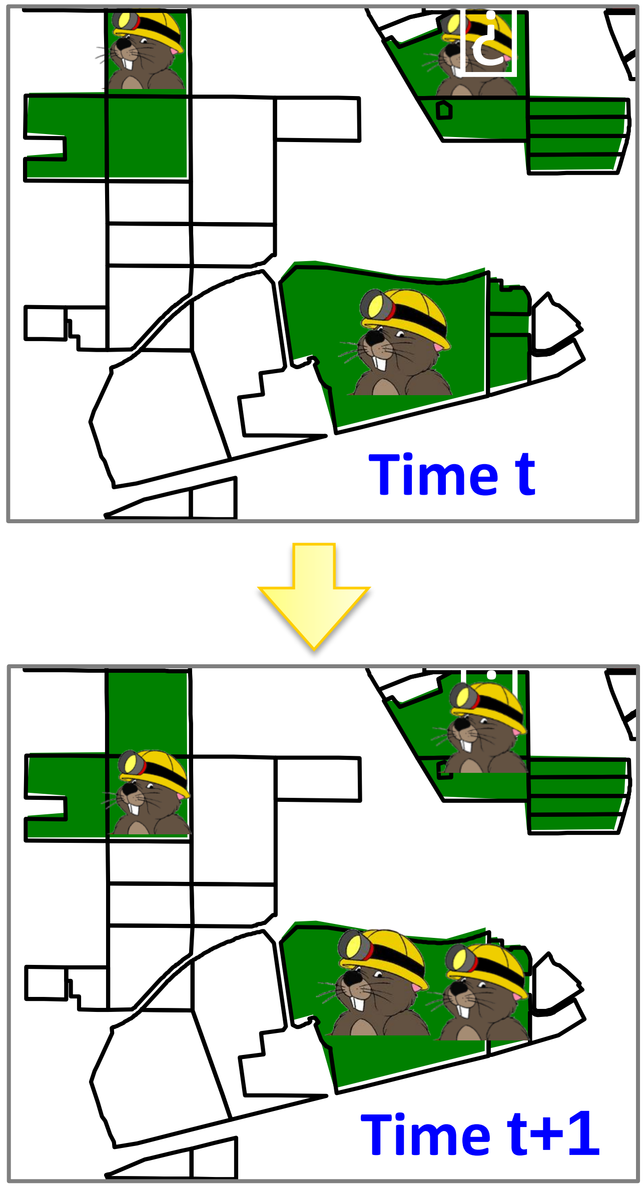
- Land parcel details
- Geography: Roads, Rivers, etc
- Model of Species' Population Dynamics
  - Reproduction, Colonization, Predation, Disease, Famine, Harsh Weather, …

Population Dynamics

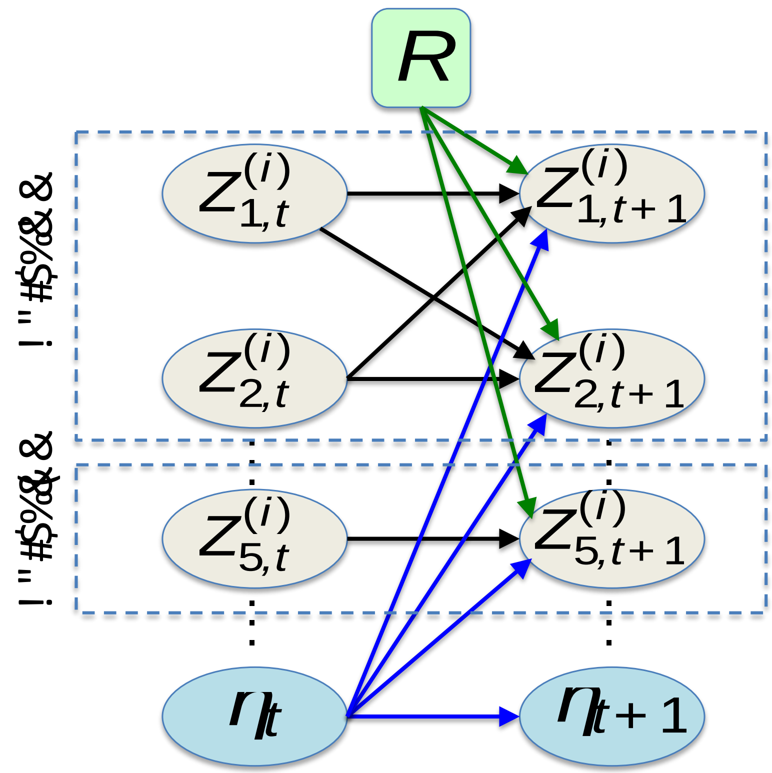
Our Choices

Protected Parcels

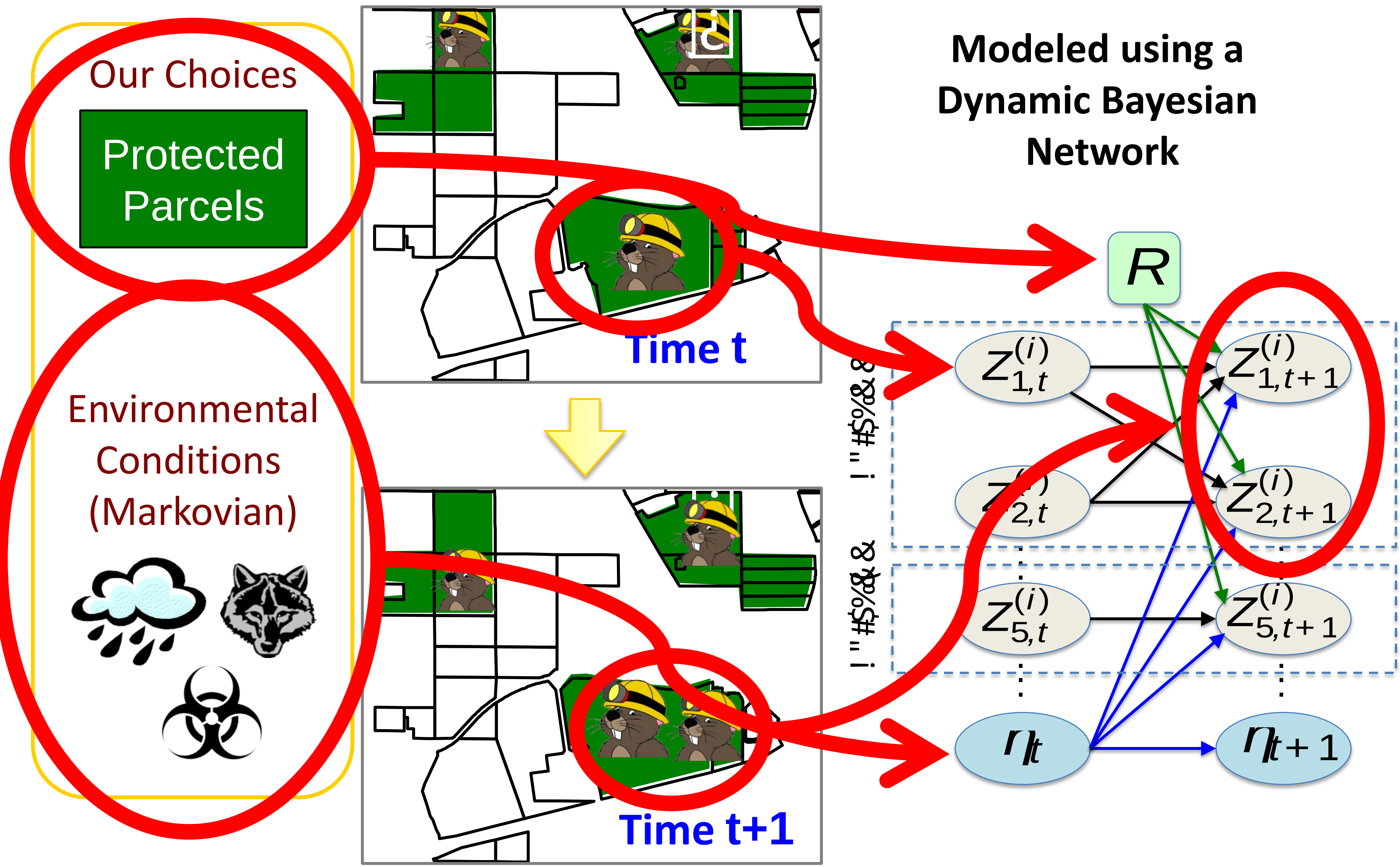
Environmental Conditions (Markovian)



Modeled using a Dynamic Bayesian Network

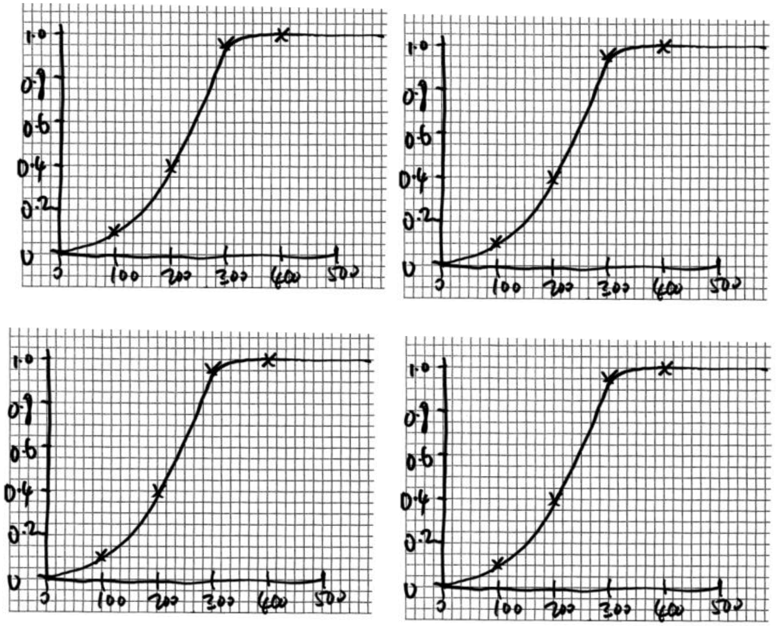


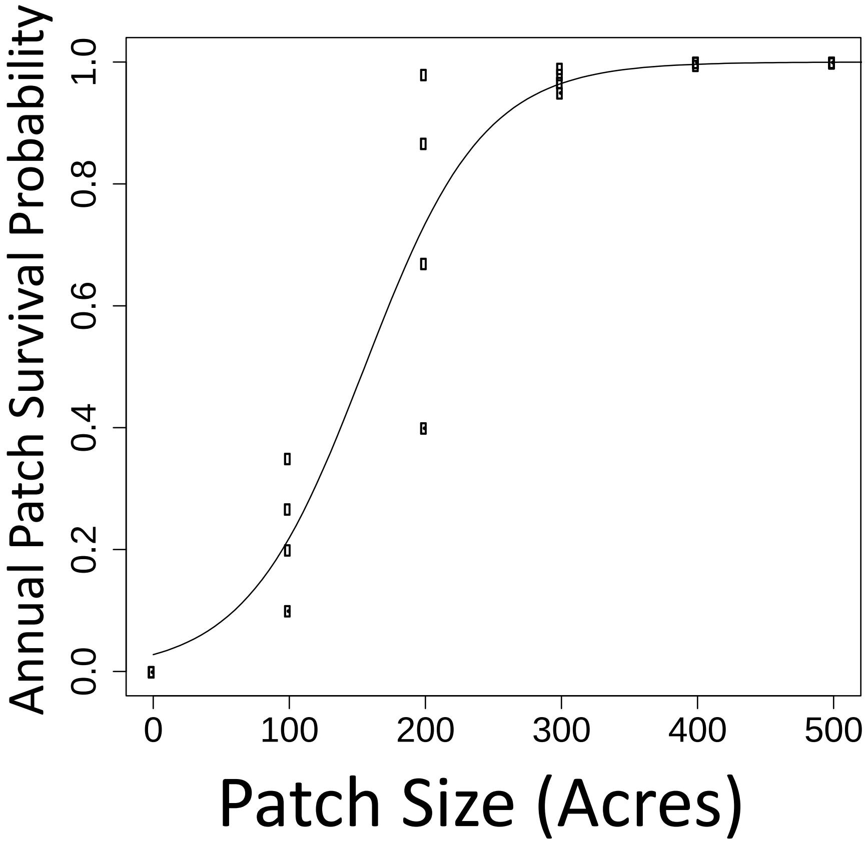
Population Dynamics



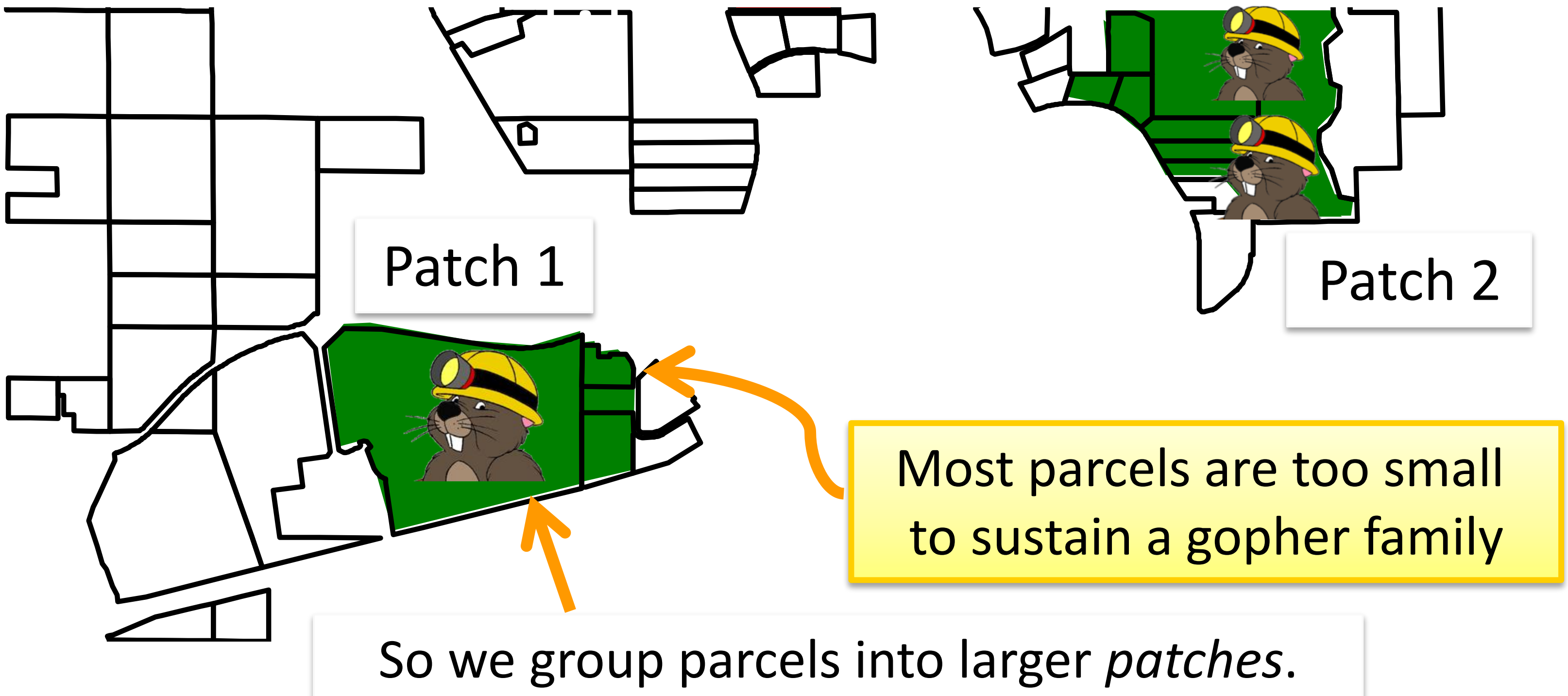
Model Parameters

- From the ecology literature, or
- Elicited from panels of domain experts



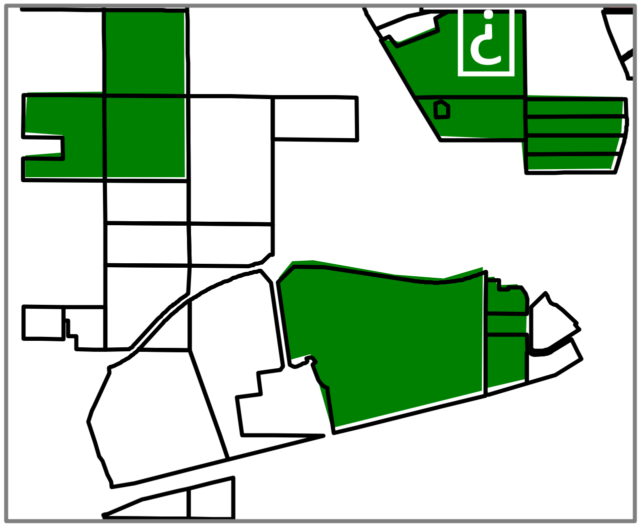


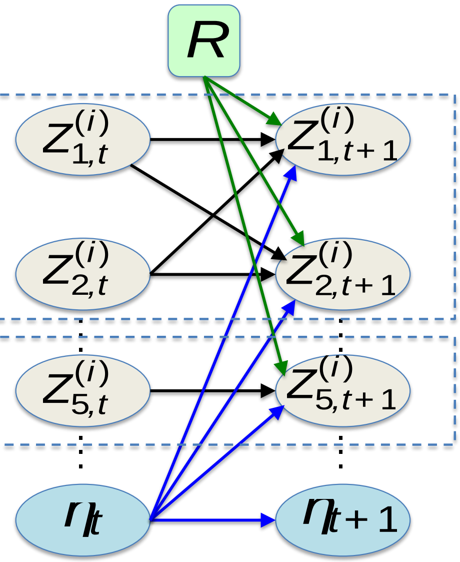
From Parcels to Patches



We assume no colonization *between* patches, and model only colonization *within* patches. We optimize over (sets of) patches.

The Objective Function





Pr[alive after 50yrs]

0.8

0.7

0.5

Selected patches R → f(R)= 2.0 (Expected # alive)

- In practice, use sample average approximation

$$f_{\mathrm{SA}} \propto \sum_{\text{simulation } \sigma} f_\sigma$$

- Choose R to maximize species persistence

"Static" Conservation Planning

- Select a reserve of maximum utility, subject to budget constraint

$$\max_R f(R) \text{ s.t. } c(R) \leq B$$

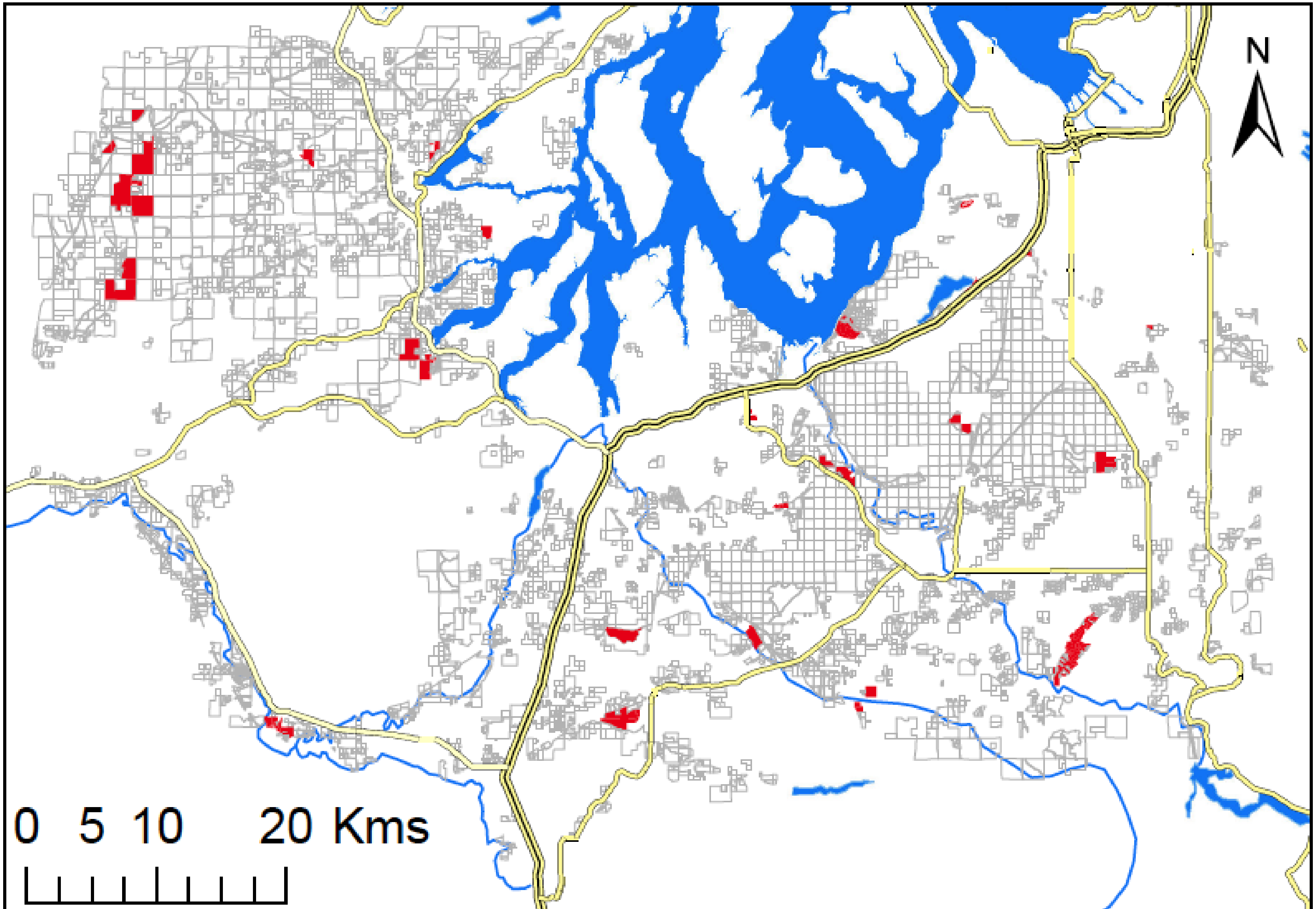
NP-hard

But f is **submodular**

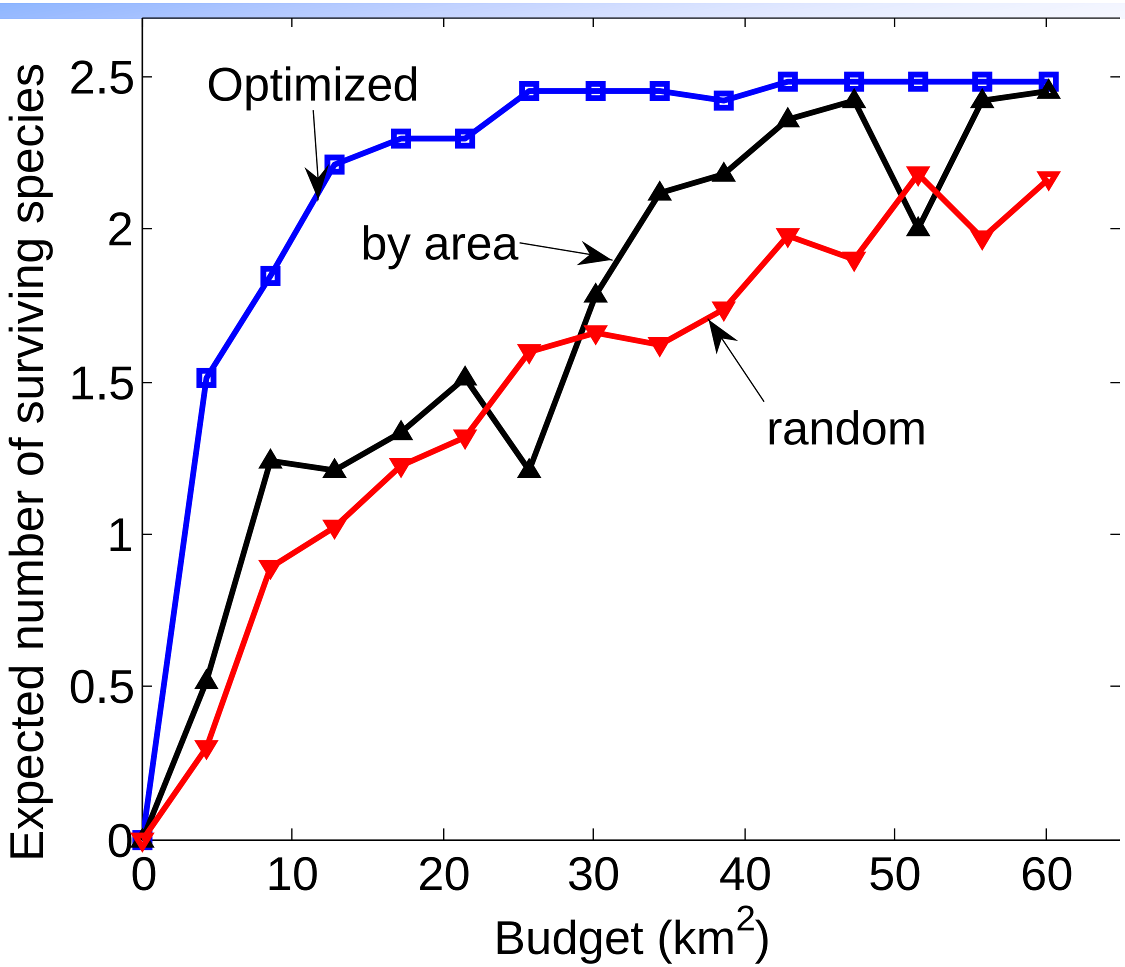
➔ Can find a near-optimal solution!

➔ Even in "incentive-compatible" manner [Singer '10]

Solution



Results: “Static” Planning



Can get large gain through optimization

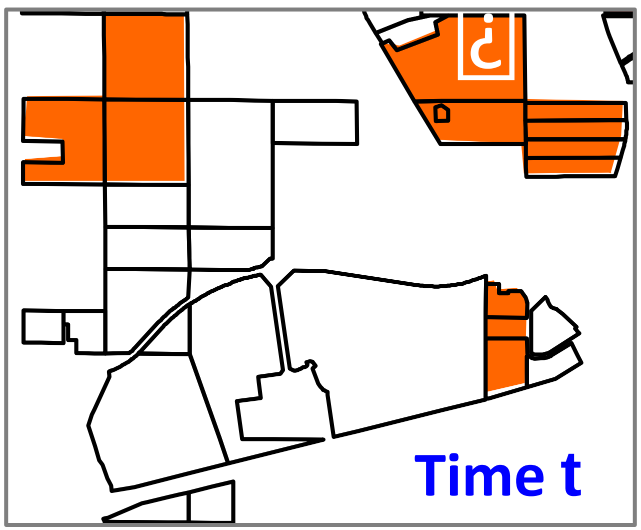
Other interesting directions

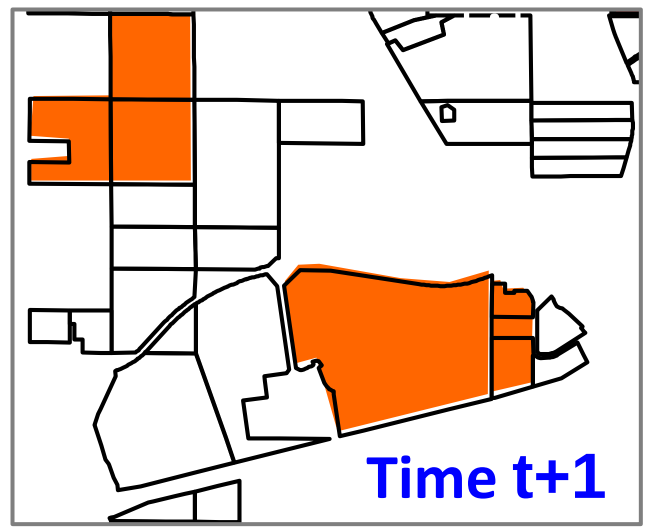
- Many sensing problems involve maximization of monotonic submodular functions
  - Can use greedy algorithm to get near-optimal solutions!
  - Lazy evaluations provide dramatic speedup

- How can we handle more complex settings:
  - Complex constraints / complex cost functions?
  - Sequential decisions?

Dynamic Conservation Planning

- Build up reserve over time
- At each time step t, the budget B_t and the set V_t of available parcels may change

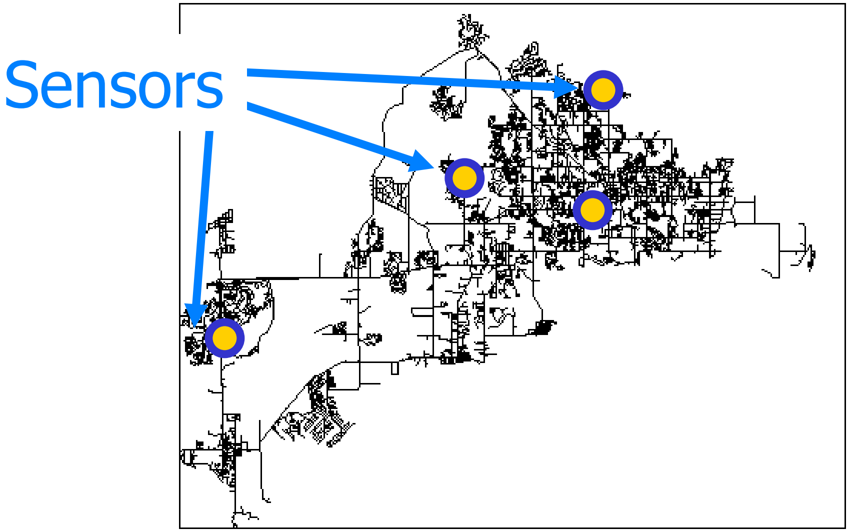




- May learn from information we gain after selecting patches

Benefit of adaptivity

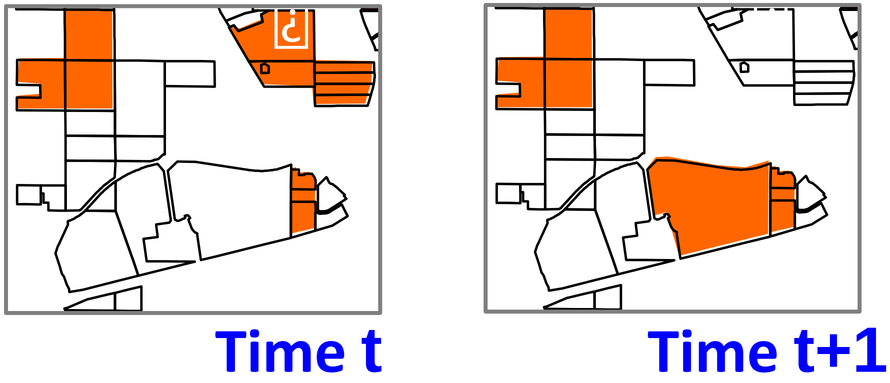
„A priori“ decisions



Find near-optimal set A of sensor locations

Must commit to all actions in advance (no „observations“)

Sequential decisions



Want near-optimal policy π for allocating resources based on observations

Is there a notion of submodularity for policies??

Problem Statement

Given:

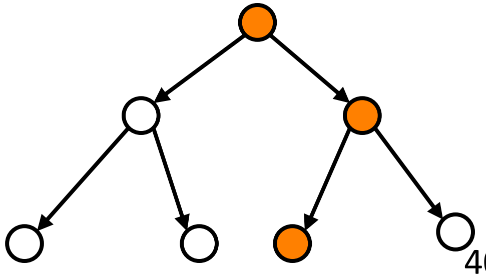
- Items (patches, tests, …) V={1,…,n}
- Associated with random variables $X_1,\ldots,X_n$ taking values in O
- Objective: $f : 2^V \times O^V \rightarrow \mathbb{R}$
- Policy π maps observation $\mathbf{x}_A$ to next item

Value of policy π: $$F(\pi) = \sum_{\mathbf{x}_V} P(\mathbf{x}_V) f(\pi(\mathbf{x}_V), \mathbf{x}_V)$$

Patches picked by π if world in state $\mathbf{x}_V$

Want $\pi^* \in \underset{|\pi| \leq k}{\operatorname{argmax}} F(\pi)$

NP-hard (also hard to approximate!)



Adaptive greedy algorithm

- Suppose we've seen $X_A = x_A$.
- Conditional expected benefit of adding item s:

$$\Delta(s \mid \mathbf{x}_A) = \mathbb{E}\Big[\underbrace{f(A \cup \{s\}, \mathbf{x}_V) - f(A, \mathbf{x}_V)}_{\text{Benefit if world in state } \mathbf{x}_V} \mid \mathbf{x}_A\Big]$$

Conditional on observations $\mathbf{x}_A$

<u>Adaptive Greedy algorithm</u>:

Start with $A = \emptyset$

For i = 1:k

- Pick $s_k \in \operatorname*{argmax}_s \Delta(s \mid \mathbf{x}_A)$
- Observe $X_{s_k} = x_{s_k}$
- Set $A \leftarrow A \cup \{s_k\}$

When does this adaptive greedy algorithm work??

Adaptive submodularity
[Golovin & Krause, JAIR 2011]

Adaptive monotonicity:

$$\Delta(s \mid \mathbf{x}_A) \geq 0$$

x_B *observes more* than x_A

Adaptive submodularity:

$$\Delta(s \mid \mathbf{x}_A) \geq \Delta(s \mid \mathbf{x}_B) \quad \text{whenever} \quad \mathbf{x}_A \preceq \mathbf{x}_B$$

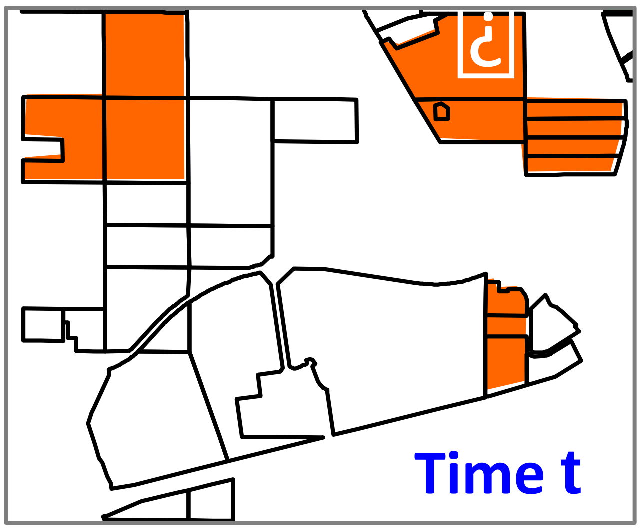
Theorem: If f is adaptive submodular and adaptive monotone w.r.t. to distribution P, then

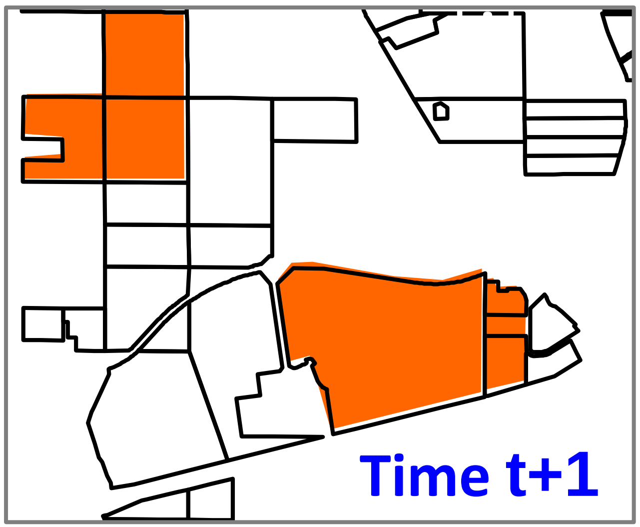
$F(\pi_{greedy}) \geq (1\text{-}1/e)\ F(\pi_{opt})$

Many other results about submodular set functions can also be "lifted" to the adaptive setting!

Dynamic Conservation Planning

- Build up reserve over time
- At each time step t, the budget B_t and the set V_t of available parcels may change



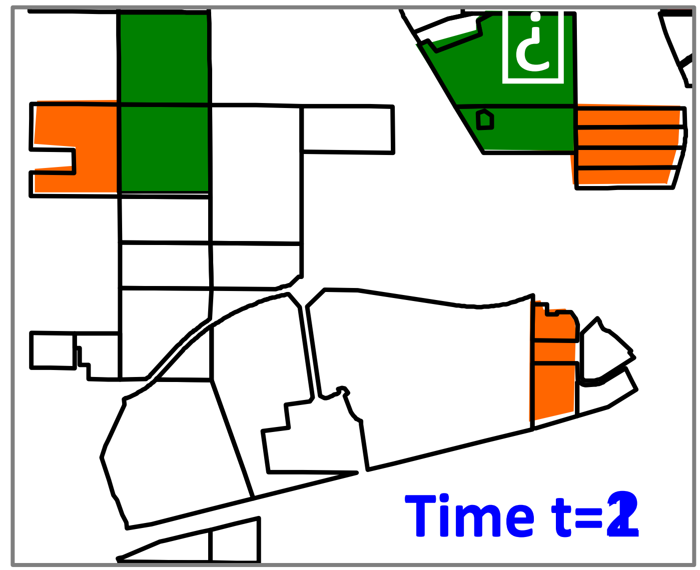


- May learn from information we gain after selecting patches
- f is **adaptive submodular** in this setting! ☺

Opportunistic Allocation for Dynamic Conservation

In each time step:

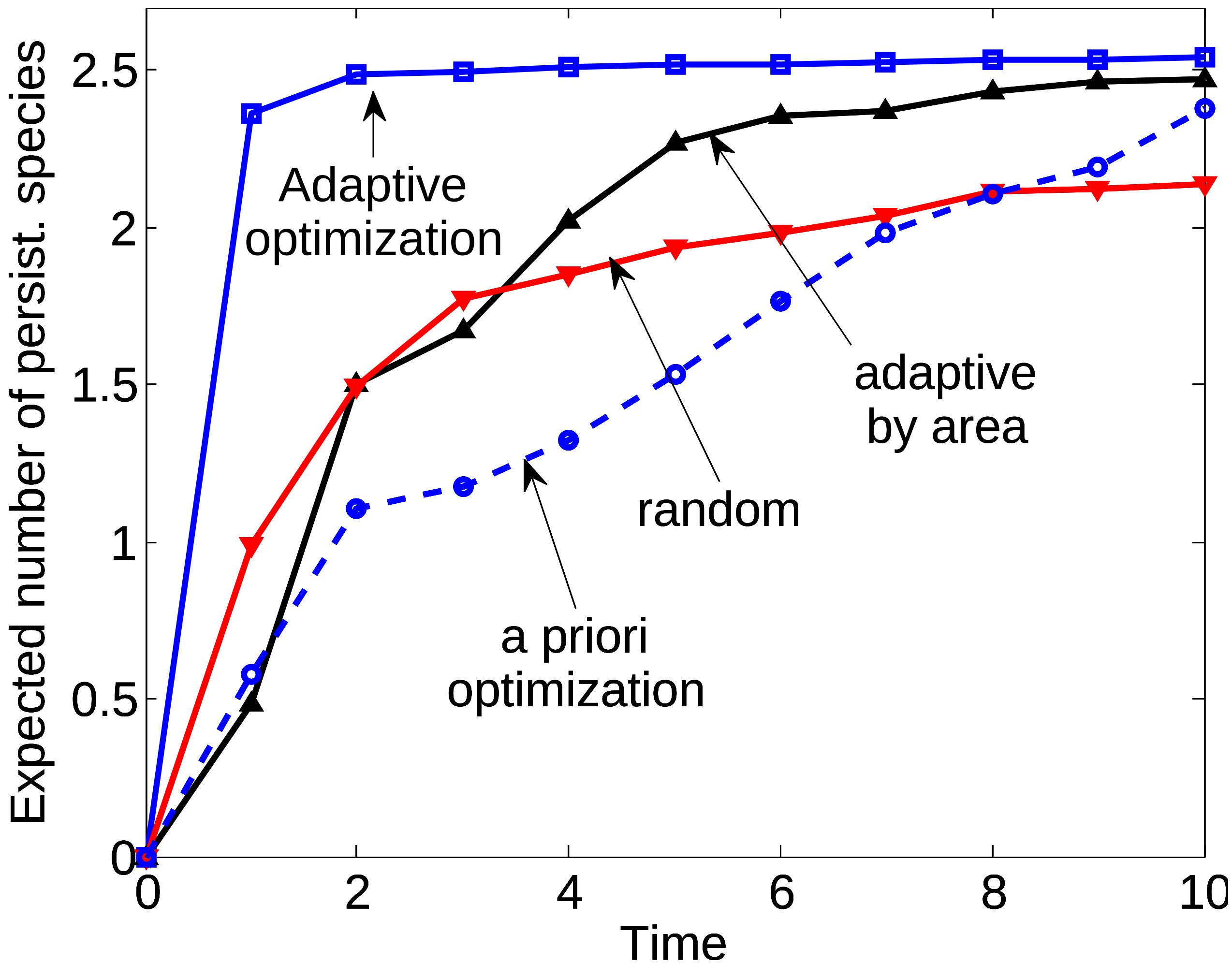
- Available parcels and budget appear
- Opportunistically choose near-optimal allocation



Theorem: We get at least 38.7% of the value of the best clairvoyant algorithm*

* Even under **adversarial** selection of available parcels & budgets!

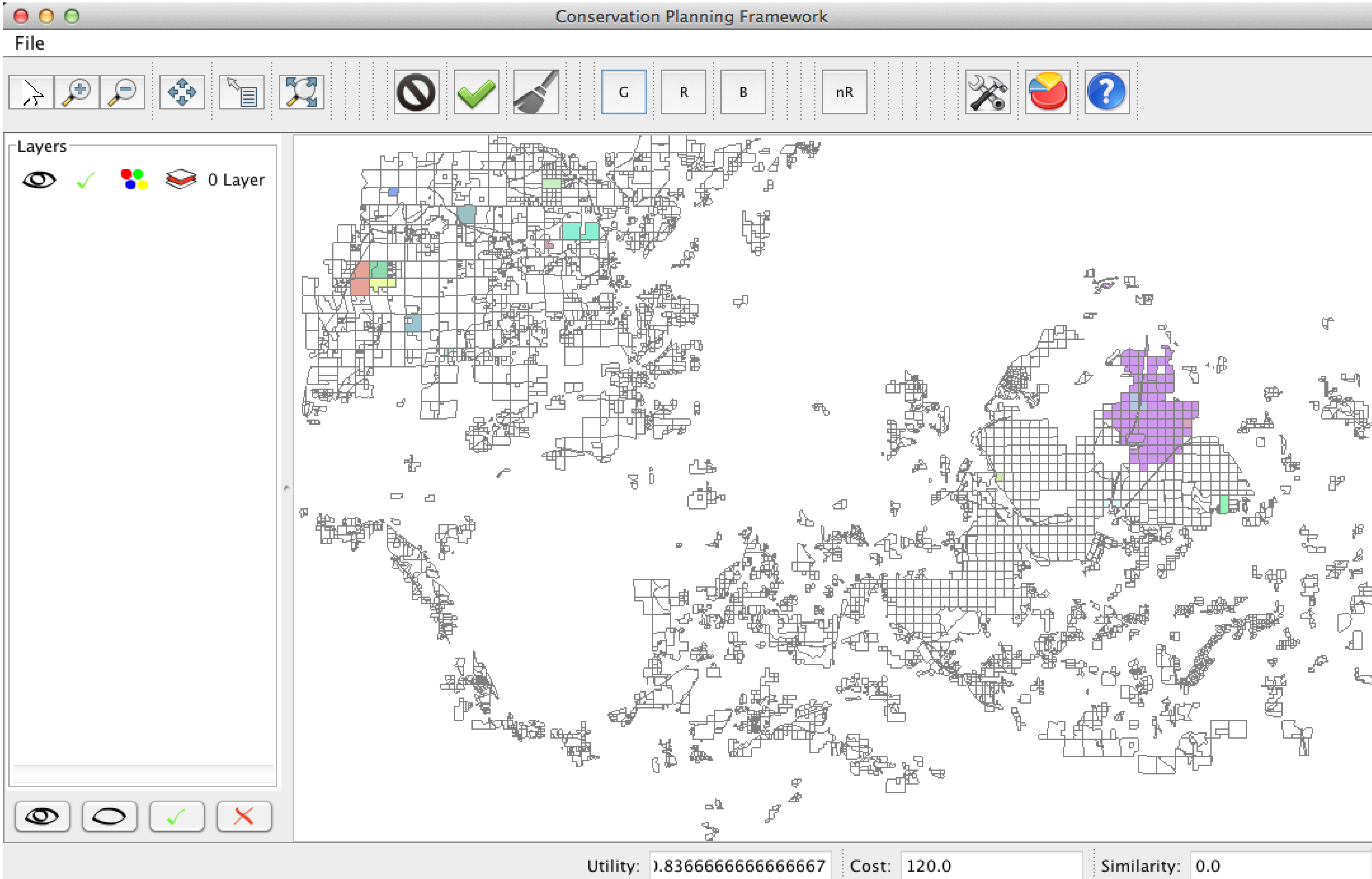
Results



Adaptive optimization outperforms existing approaches

Decision-Support Tool

[w Bogunovic, Converse]



Near real-time, interactive solver, see talk tomorrow!

Related Work

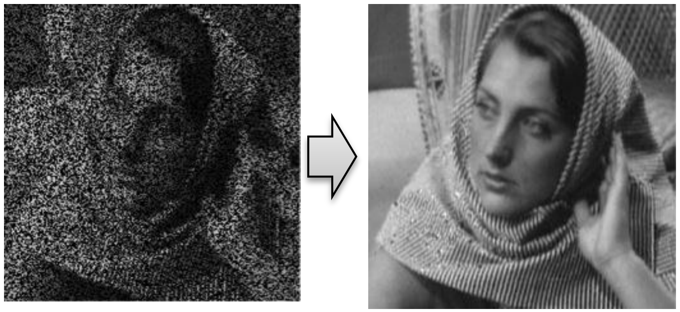
- Existing software
  - Marxan [Ball, Possingham & Watts '09]
  - Zonation [Moilanen and Kujala '08]
  - General purpose software
  - No population dynamics modeling, no guarantees
- Sheldon et al. '10
  - Models non-submodular population dynamics
  - Only considers static problem
  - Relies on mixed integer programming

Other applications of Adaptive Submodularity

- Stochastic set cover
- Active learning
- Bayesian experimental design / value of information
- Influence maximization in social networks
- ...

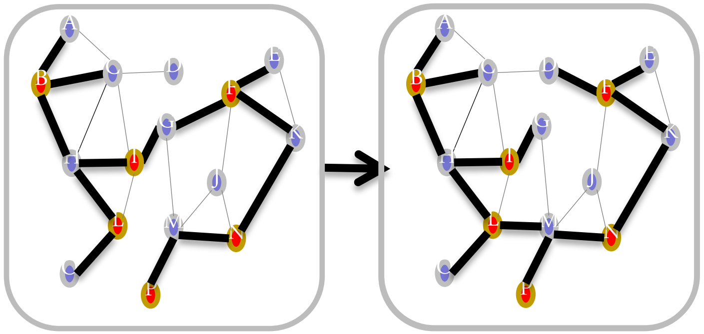
- **Submodular surrogates?**

Submodularity in ML / AI

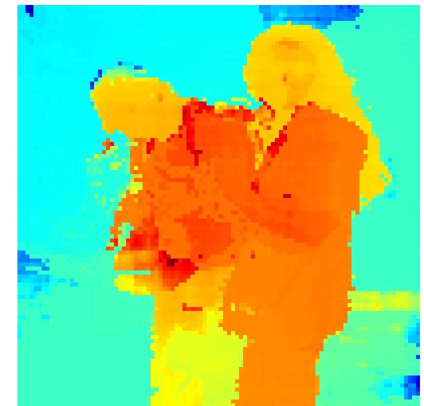


$$\mathbf{x} = \mathbf{\Phi}\mathbf{w} + \mathbf{n}$$

Submodular dictionary selection for sparse representation [ICML '10]

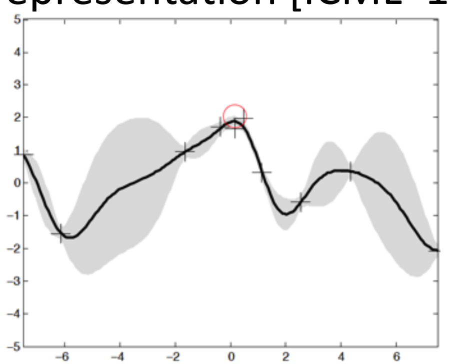


Submodular compressive sensing [AISTATS '12]



$$\max_{\mathbf{x}} P(\mathbf{x} \mid \mathbf{y})$$

Fast inference for high-order submodular MAP problems [NIPS '10]



$$R_T = \mathcal{O}^*\left(\sqrt{T \max_{|\mathcal{A}| \le T} F(\mathcal{A})}\right)$$

First regret bounds for GP optimization [NIPS '11]

- MATLAB Toolbox for optimizing submodular functions (JMLR '10)
- Series of NIPS Workshops on Discrete Optimization in ML
  ➔ videos on *videolectures.net*

Conclusions

- Many applications in computational sustainabiity need large-scale discrete optimization under uncertainty
- Fortunately, some of those have structure: submodularity
- Submodularity can be exploited to develop efficient, scalable algorithms with strong guarantees
- Can handle complex constraints
- Adaptive submodularity allows to address sequential decision problems

Thanks: