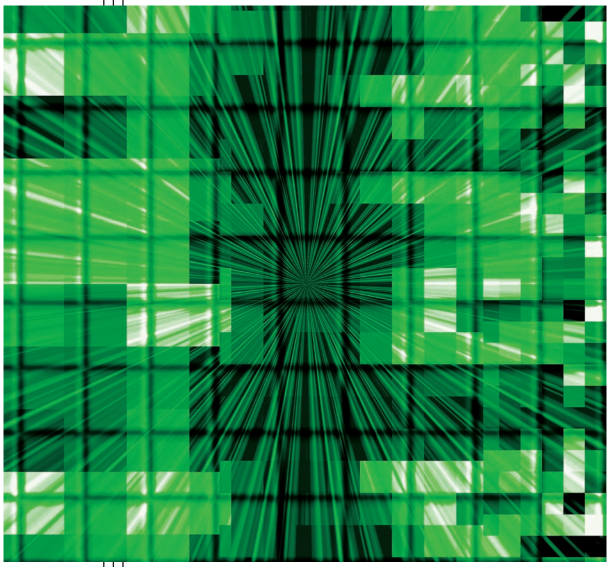

OPTIMIZING SENSING: FROM WATER TO THE WEB

Andreas Krause, *California Institute of Technology*
Carlos Guestrin, *Carnegie Mellon University*

Where should we place sensors to quickly detect contamination in drinking water distribution networks? Which blogs should we read to learn about the biggest stories on the Web? Such problems are typically NP-hard in theory and extremely challenging in practice. The authors present algorithms that exploit submodularity to efficiently find provably near-optimal solutions to large, complex real-world sensing problems.

High levels of pollutants, such as nitrates, in lakes and rivers can lead to the rapid growth of algae. These algal blooms can be a severe threat to our drinking water resources. For example, the algal bloom at Taihu Lake, Jiangsu Province, China, in June 2007 deprived four million people of drinking water and required an estimated $14.5 billion in cleaning costs (www.msnbc.msn.com/id/21498294/, June 2007).

Many of the growth processes associated with such algal blooms are still not sufficiently well understood and need to be studied in lakes instead of the lab. A natural approach to obtaining this understanding is to monitor environmental factors such as temperature, nutrient distribution, and fluorescence that are associated with such algal blooms. A promising technology for this goal uses sensors carried by robotic boats to obtain measurements over a large spatial area. However, since making each measurement requires a significant amount of time, and the robot needs a certain amount of time to travel between sensing locations, it is important to plan trajectories that allow obtaining the most useful information about the phenomenon under study.[1]

A similar problem arises in the context of monitoring municipal drinking water distribution networks. Accidental or malicious contamination of such networks can affect a large population and cause significant harm. We can potentially detect such contamination by deploying sensors in the water distribution network. However, these sensors are fairly expensive—for example, the Hach Water Distribution Monitoring system is currently priced above $13,000 per unit. Thus, we must optimize the locations where these costly sensors are placed to maximize their effectiveness. The need to address this task has received significant attention, and recently the Battle of the Water Sensor Networks was instituted as a benchmark challenge to encourage research in this area.[2]

More generally, the goal in sensing optimization is to learn something about the state of the world (such as temperature at a given location or whether there is contamination) by optimally making a small set of expensive measurements. The fundamental question is, How can we get the most useful information at minimum cost? This problem has been studied in several disciplines, including statistics (experimental design), machine learning (active learning), operations research (facility location), sensor networks, and robotics. However, most of the existing approaches rely on heuristics without guarantees, which can potentially perform poorly. There are also algorithms that are designed to find the optimal solution, such as algorithms for solving partially observable Markov decision processes (POMDPs) or mixed-integer programs (MIPs). However, it is often very difficult to scale these techniques to large problems.

We describe a new class of algorithms that have strong theoretical performance guarantees and scale to large sensing problems. Our algorithms are based on the key insight that many sensing problems satisfy *submodularity*, an intuitive diminishing-returns property: Adding an observation helps more if we have made few observations than if we already have made many observations. When a problem satisfies this diminishing-returns property, we can develop effective algorithms that are guaranteed to both perform well in theory and work well in practice.

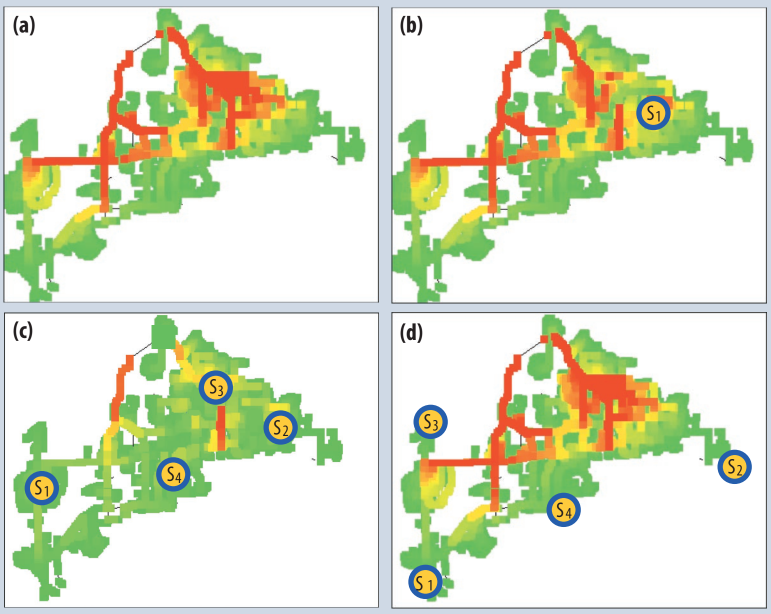

Figure 1. Impact of contamination events on a large drinking water distribution network. The color of each node indicates the impact severity before detection if a contaminant is introduced at that node and sensors are placed at the indicated locations. Red indicates high impact, green indicates low impact. In (a), no sensors are placed, hence contamination is never detected; (b) results after placing one sensor; (c) shows effective placement; and (d) shows poor placement.

SENSING OPTIMIZATION PROBLEMS

The sensing optimization problem seeks to find a set of sensors that provides the best possible sensing quality at the minimum possible cost. Let's start by defining the notions of sensing quality and cost. Intuitively, a sensing quality function $F(A)$ provides a score for selecting the set of locations A out of the possible locations V.

Consider the problem of protecting our water distribution systems from contamination. Here, $F(A)$ measures, for example, the number of people protected by placing sensors at locations A. To understand such a function better, consider the effect of introducing a contaminant at some location i; as this contaminant spreads through the network, thousands or more people may be affected. Due to the system's complex dynamics, contamination at some locations spreads farther than at others, as Figure 1a shows. Once we place a sensor, some contamination is detected earlier, as shown in Figure 1b, and more people are protected from the contaminant. A good set of sensor placements A, shown in Figure 1c, will detect contamination early, so $F(A)$ will be high. Conversely, poor placement A', as shown in Figure 1d, will have low $F(A')$.

In general, the goal of sensing optimization is to find a set A^* that provides the maximum possible value. Intuitively, we can maximize $F(A)$ by placing sensors at every possible location, but in real applications there are resource constraints in that sensors cost money and budgets are limited.

The simplest type of constraint is a cardinality constraint: We have k sensors, and thus want to ensure that $|A| \le k$. In this case, the optimization problem becomes

$$A^* = \operatorname{argmax}_A F(A) \quad (1)$$
$$A \subset V: |A| \le k.$$

More generally, sensors can have different costs—for example, drilling into a pipe may be more expensive than placing a sensor in an existing junction. Here, we denote the cost of sensor s by $C(s)$; the cost of a set of sensors A is the sum of their individual costs, $C(A) = \sum_{s \in A} C(s)$. If we have a maximum budget B to spend, then our constraint becomes $C(A) \le B$.

SUBMODULARITY

Even the simplest formulation of these sensing problems in Equation 1 is already NP-hard,[3] and it is unlikely that there will be an efficient algorithm for solving this problem exactly. However, consider what perhaps is the simplest possible heuristic: the *greedy algorithm*. This procedure starts by picking the element s_1 that provides the most information: $s_1 = \operatorname{argmax}_s F(s)$. Then, we iteratively pick elements s_i that provide the most additional information: $s_i = \operatorname{argmax}_s F(\{s\} \cup \{s_1, \ldots, s_{i-1}\})$.

This heuristic seems naïve, because, for example, when we pick the second sensor, the choice of the first sensor is

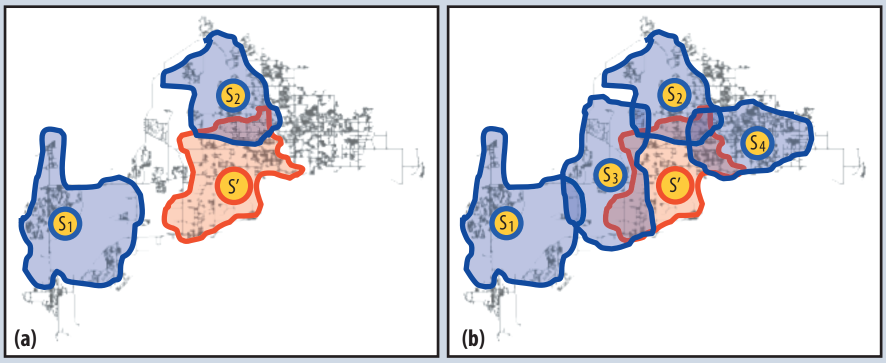

Figure 2. Illustration of the diminishing-returns effect (submodularity) of adding sensors. The blue regions indicate nodes where contamination is detected quickly using the existing sensors. The red region indicates the additional coverage by adding a new sensor s. If more sensors are already placed (b), there is more overlap, hence less gain in sensing quality.

fixed and cannot be revised. This simple heuristic, however, performs surprisingly well in practice in many real-world applications. In fact, for many practical applications it is hard to find an algorithm that performs significantly better than the greedy approach for the optimization problem in Equation 1. This empirical observation leads to an interesting theoretical question: Why does the greedy algorithm perform so well on sensor placement problems?

This question can be answered by introducing *submodularity*, a structural property that is present in many practical sensing optimization problems. Intuitively, a problem is submodular if we observe *diminishing returns*. For example, if we have deployed five sensors, a sixth one will provide much additional information, whereas after deploying 500 sensors the 501st will provide much less new information. Figure 2 illustrates this concept using the classic notion of set cover. Diminishing returns can be seen throughout our lives—for example, when applied to business practices, where the Pareto principle says that "80 percent of your sales come from 20 percent of your clients." In water distribution systems, we naturally see diminishing returns: A few sensors may protect a large portion of the population, but to protect the last few people we may need many more sensors.

More formally, a set function is said to be submodular[4] if adding an element s to a set A provides more gain than adding it to a set B; in other words, if A is a subset of B—that is, $\forall A \subset B \subseteq V$ and $\forall\ s \in V \setminus B$—we have $F(\{s\} \cup A) - F(A) \geq F(\{s\} \cup B) - F(B)$. Many real-world problems satisfy this property—for example, the function measuring the population protected by sensing in water distribution systems.[3]

If a problem is submodular, we can exploit this property to develop efficient algorithms that are guaranteed in theory to provide near-optimal solutions and that perform extremely well in practice. In particular, for the case of placing k sensors in submodular problems, the greedy algorithm is guaranteed to provide near-optimal solutions, thus providing a theoretical explanation for the strong empirical performance of this simple heuristic.

PHYSICAL SENSING CASE STUDY

Accidental and malicious contamination in municipal drinking water distribution networks can pose a severe threat to the population. Such contamination could potentially be detected by deploying sensors in the network's junctions and pipes, but because existing sensor solutions are fairly expensive, only small numbers of these sensors can be deployed. The problem of deploying a small number of sensors to optimize performance criteria such as time to detection or minimizing the expected population affected by contamination has received significant attention.

During the 8th Annual Water Distribution Systems Analysis Symposium in August 2006, the Battle of the Water Sensor Networks was organized. The 13 teams participating in the BWSN challenge competed for optimal performance.[2] The challenge included a realistic model of a metropolitan-area water-distribution network with 12,527 nodes, as well as a description of 3.6 million realistic contamination scenarios, which varied in the choice of injection location and time of introduction of the contaminant into the network, as well as other parameters. The EPANET 2.0 simulator developed by the US Environmental Protection Agency[5] was used to estimate the impact of possible contamination. The goal of the challenge was to optimize sensor deployments with respect to multiple objectives: minimizing the time to detection, the expected population affected by contamination, and the total amount of contaminated water consumed, and maximizing the detection likelihood.

More formally, for a set A of sensor locations, the resulting value $F(A)$ will be:

$$F(A) = \sum_{i \in I} p(i) \max_{s \in A} M_{is}, \tag{2}$$

where M_{is} is, for example, the population protected by early detection from a sensor placed at node b if a contamination originates at node i. Objective functions of this form and the ones used in the challenge are submodular.[3] Our approach was to exploit this submodularity property to provide near-optimal sensor placements. However, evaluating the objective function $F(A)$ requires the values M_{is}, and thus the estimation of the impact of all 3.6 million contamination events $i \in I$, along with the benefit of placing each sensor at each possible location.

Due to the magnitude of the problem in the challenge (the largest water distribution network studied by the community at that time), this task required massive amounts of distributed computation, processing approximately 47 terabytes of simulation data in a cluster of 20 multicore machines.[3] In addition, evaluating the function $F(A)$ is very computationally expensive, and running the greedy algorithm to select 20 sensors takes approximately 30 hours in a highly optimized implementation. By exploiting submodularity, we drastically reduced this computation time to approximately one hour using the *lazy evaluations technique*, while retaining the strong theoretical guarantees.[3] We call this the cost-effective lazy forward-selection (CELF) algorithm.

To evaluate the entries in the challenge, parameter settings were varied in the problem instances, including the amount of contaminant introduced and the delay before detection. Altogether, contributions were evaluated in 30 different settings. Since the goal was to solve a multicriteria optimization problem, for each of these settings the set of nondominated entries was identified. (An entry was considered to dominate even if there were other entries that performed at least as well in all objectives, but the entry in question performed strictly better in one of the objectives.) Each participating research team was evaluated based on the total number of nondominated solutions (with 30 being the maximum possible score).

The participants in the challenge used different algorithms for optimization, including genetic algorithms and other heuristics and exact solvers (such as approaches based on MIPs) that could only be applied to smaller parts of the problem instance due to its size. Some participants did not use algorithms, but rather their domain knowledge and prior expertise to make the decision. According to the performance evaluation by the organizers of the challenge, our solution based on submodular function optimization obtained a 24 percent higher score than the runner-up.[2]

Robust sensing optimization

When applied to a water distribution network, the CELF algorithm selects sensor locations to detect random, accidental contamination events. However, an adversary who learns about the sensor locations can act strategically and maliciously and contaminate the network based on that knowledge. To protect against such an adversary, it is important to decide where to place sensors to maximize the worst-case detection performance.

This and many other problems can be formulated as optimizing a sensor deployment against an adversary who can, after seeing the deployment, select the minimum over a set of submodular functions. More formally, we pick A* to maximize

$$A^* = \underset{A \subset V : |A| \le k}{\arg\max} \; \min_i \; F_i\,(A), \tag{3}$$

where $F_1, \ldots, F_m$ is a collection of monotonic submodular functions. In the water networks example, F_i is the detection performance that sensors A achieve if the adversary chooses contamination scenario $i \in I$. In the following, we describe an algorithm for solving such robust sensing problems.

The Saturate algorithm reduces the nonsubmodular, worst-case objective to a submodular optimization problem that can be approximately solved using the greedy algorithm.

Given the provably near-optimal performance of the greedy algorithm for optimizing submodular functions—that is, for solving Equation 1—a natural approach to the robust optimization Equation 2 would be to modify the greedy algorithm so that after elements $s_1, \ldots, s_{j-1}$ have been selected, it adds the element

$$s_j = \underset{s \in V}{\arg\max} \min_i \; F_i(\{s_1, \ldots, s_{j-1}\} \cup \{s\}).$$

For example, in the water network example, it adds the sensor that most decreases the worst-case detection time.

Unfortunately, for the robust optimization Equation 2, this greedy algorithm can arbitrarily fail badly.[6] To see this, consider the following example. Suppose there are two contamination events, i_1, i_2, and three possible sensor locations, s_1, s_2, and s_3. A sensor at location s_1 can immediately detect event i_1 but never detect i_2; similarly, s_2 can immediately detect i_2 but never detect i_1. A sensor at location s_3, however, can detect both contamination events, but only after a long time.

Now suppose we can afford to place two sensors. If we run the greedy algorithm, it will place a sensor at location s_3 first, since that is the only location that can detect both events. But once the greedy algorithm commits to picking s_3, it can only select s_1 or s_2, but not both, leaving one of the contamination events essentially unprotected. On the other hand, the optimal choice is to pick s_1 and s_2, thus detecting both contaminations. This example illustrates the fact that the simple greedy algorithm that was useful before will perform very badly when it comes to problems that require robustness.

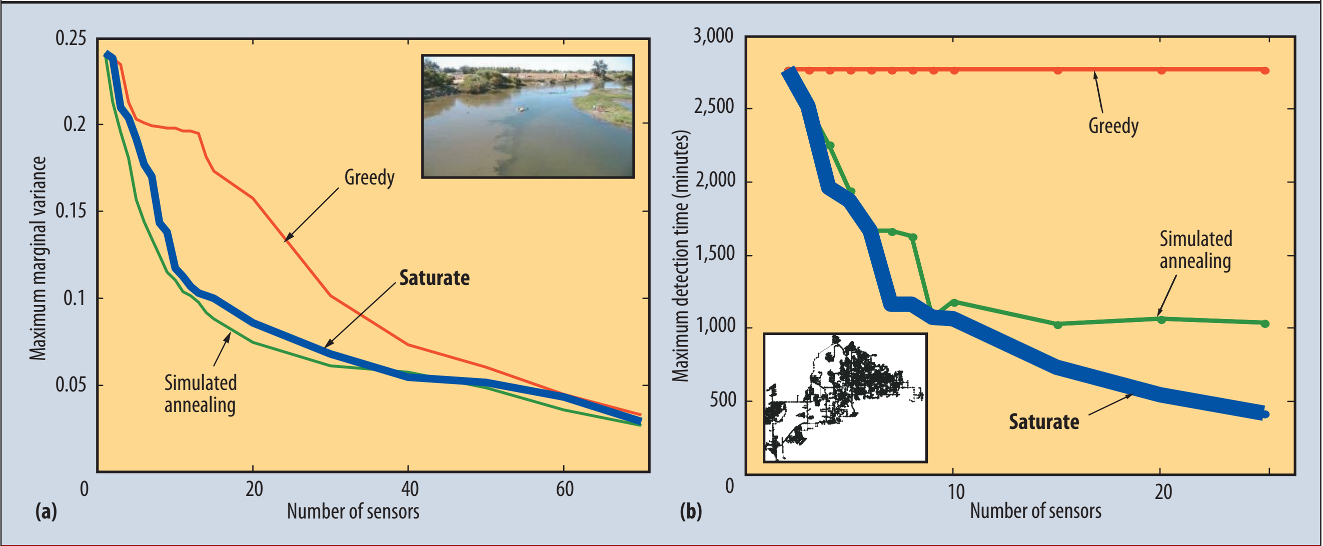

Figure 3. Performance of Saturate for (a) environmental monitoring and (b) water networks. Saturate dramatically outperforms the greedy algorithm and is competitive with or superior to a highly fine-tuned simulated annealing algorithm.

Saturate algorithm

Motivated by the poor performance of the greedy algorithm, we developed Saturate, a novel algorithm for optimizing the worst-case detection performance. The key idea behind Saturate is to reduce the nonsubmodular, worst-case objective to a submodular optimization problem that can be approximately solved using the greedy algorithm. This transformation is achieved by first "guessing" the value c of the optimal solution (this guessing is implemented as a binary search). Then, the greedy algorithm is applied to "saturate" all objectives (that is, ensure that $F_i(A) \geqq c$). This reformulation of the problem allowed us to design this very simple algorithm that is theoretically guaranteed to obtain a near-optimal solution to the robust sensing problem.

We have evaluated our Saturate algorithm on several real-world sensing problems. The first application is in environmental monitoring. Problems such as monitoring algal blooms in lakes require estimating spatial phenomena such as temperature, fluorescence, and nutrient distribution over a large spatial area. One approach to estimate a phenomenon such as temperature is to make measurements at a small number of locations (for example, using sensor buoys or sensors carried by robotic boats) and then predict the temperature at the unobserved locations using statistical models. Gaussian processes (also known as *kriging* models) have been found effective for this purpose. However, to obtain accurate predictions, we must select measurement locations that minimize the expected error in the predictions made given these measurements.

One approach to quantify the prediction error at some location $s \in V$ is the expected posterior variance $\mathrm{Var}(X_s \mid X_A)$ given that we have made measurements at locations A. In many cases, the reduction in variance at location s, $F_s(A) = \mathrm{Var}(X_s) - \mathrm{Var}(X_s \mid X_A)$ can be shown to be a submodular function.[7] Optimizing the average prediction error throughout the environment (for example, a lake) is a natural (submodular) function we can optimize. Unfortunately, such an average-case optimization does not guarantee good predictions at every point in the lake, which could result in missing an important event. A good alternative is to optimize the worst-case prediction error over the entire lake, a problem often called *minimax kriging*. Thus, selecting a set of locations to measure to minimize the worst-case prediction error is an example of a robust submodular sensing problem.

We use the Saturate algorithm on this problem and compare it against the state of the art from geostatistics, a simulated annealing algorithm with seven parameters that need to be carefully fine-tuned for the particular application at hand. Figure 3a compares the performance of the algorithms on a problem of predicting pH values in a lake near Merced, California. We can see that the greedy algorithm performs very poorly on this task. The simulated annealing algorithm performs much better, requiring approximately half as many samples to achieve the same worst-case error. The Saturate algorithm is competitive with the state-of-the-art geostatistics algorithm, while being much simpler to implement, requiring no parameters to tune and running approximately 10 times faster in our experiment.

Note however that, when predicting these pH values, the performance of the greedy algorithm was "reasonable." We obtained a very different outcome when evaluating the algorithms on the problem of deploying sensors in drinking water distribution networks to minimize the worst-case time to detection. Figure 3b shows the results of this experiment. Interestingly, in this domain, the greedy algorithm

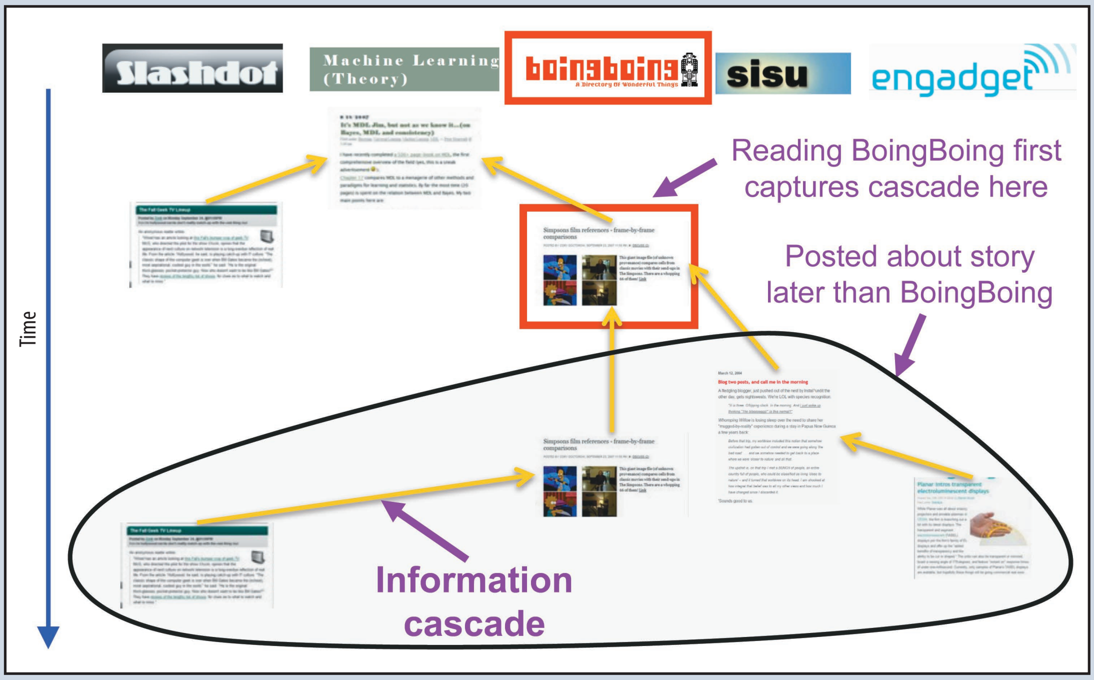
Slashdot
Machine Learning (Theory)
boingboing
sisu
engadget
Reading BoingBoing first captures cascade here
Posted about story later than BoingBoing
Time
Information cascade

Figure 4. An information cascade. Information has spread through the blogosphere at a tremendous rate, with tens of millions of active blogs generating hundreds of millions of postings per year.

does not decrease the worst-case detection time, exhibiting, in practice, the arbitrarily bad performance the greedy approach suggested by the theory. The explanation for this poor performance is that unless all contamination events are detected, the worst-case detection time remains the same. However, no single sensor can detect all contamination events; thus, the greedy algorithm has no indication of which first sensor location to pick.

The simulated annealing algorithm randomizes, so it eventually obtains nontrivial, but still poor, performance. Our Saturate algorithm exhibits much better performance in practice, obtaining 60 percent lower worst-case detection time than simulated annealing when placing 25 sensors.

FROM WATER TO THE WEB: WHAT BLOGS SHOULD I READ?

The case studies we have used thus far focus on optimization problems related to physical sensing. Sensing on the Web represents a very different problem. In 2008, *Time* asked, "How many blogs does the world need?" claiming that there were already too many out there.[8] The blogosphere has grown at a tremendous rate, with tens of millions of active blogs generating hundreds of millions of postings per year (http://spinn3r.com). This activity leads to a huge overload of information, opening up significant sensing optimization questions such as, If you only have 10 minutes a day to spend on the blogosphere, which blogs should you read to keep track of the big stories?

To understand how to address this question, consider how information spreads through the blogosphere. As illustrated in Figure 4, a story posted in a blog may be picked up (and linked to) by other blogs; from there, these postings may be linked to by yet more blogs, and so on. Such spread of information in the blogosphere forms what we call an *information cascade*.[9] A good blog captures big stories—that is, large cascades—early on, as close to the first source as possible.

At first, the problem of capturing information cascades seems quite different from the previous tasks. In reality, however, the spread of contaminants through water distribution systems is very similar to the spread of information through the blogosphere. And, most importantly, both of these tasks can be formulated as submodular optimization problems, which can be tackled by the algorithms we have described.

As an example, let us formalize one criterion for characterizing how well a blog captures a cascade c sparked by a certain story. Consider reading a particular blog b, which discussed this story at a certain time t. If blog b captures the story early on in cascade c, and then the story becomes huge, many other postings will appear in cascade c at a time later than t. In this case, blog b was very successful at capturing cascade c. On the other hand, if a story does not become popular, its cascade c' will be small. If blog b captures cascade c' late, few blogs will report on this story after blog b, but nothing much will

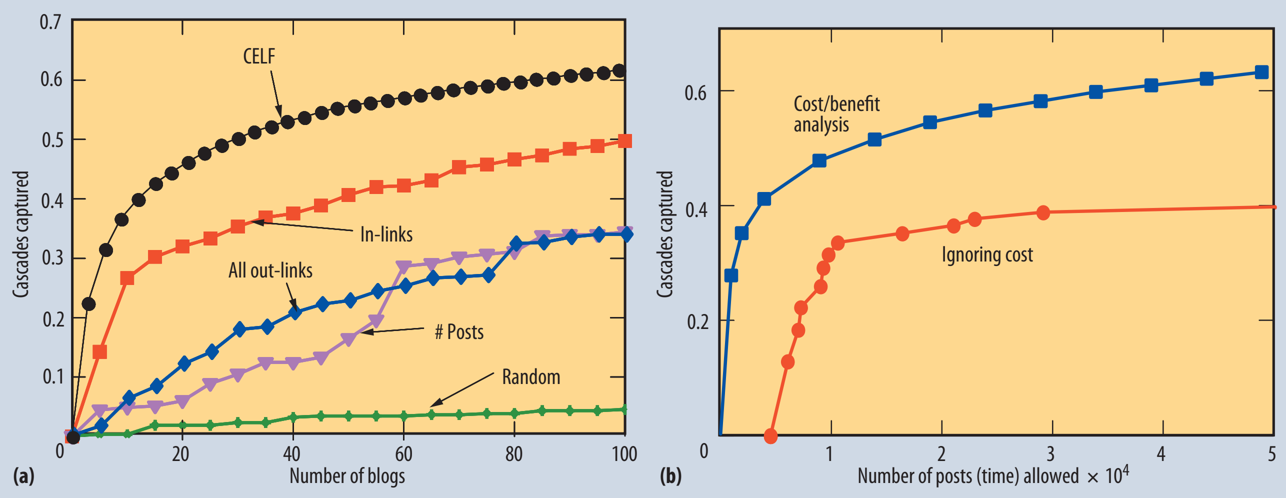

Figure 5. Performance of CELF algorithm on blog selection problem: (a) CELF outperforms heuristics; (b) cost-benefit optimization.

be lost by being late since the story was insignificant in the first place.

More formally, if we recommend a set A of blogs, the resulting value $F(A)$ is given by Equation 2, where M_{ib} is the number of postings on cascade i that are dated later than the time blog b first appears in this cascade. Thus, both this objective and the objective for water distribution systems are submodular functions and amenable to the same algorithms.

To demonstrate our algorithms in practice, consider a blog dataset from 2006, with 45,000 blogs, 10.5 million posts, and 16.2 million links (30 Gbytes of data).[9] In this dataset, we discovered 17,589 significant cascades, where each blog participated in 9.4 different cascades on average. As Figure 5a shows, the CELF algorithm greatly outperforms several other heuristic selection techniques. The best runner-up, performing about 45 percent worse than CELF, picks blogs by the number of in- or out-links from or to other blogs in the dataset, which is the type of heuristic many blog search websites use.

The results presented so far assume that every blog has the same cost. Under this unit cost model, the algorithm tends to pick large, influential blogs that have many posts. For example, instapundit.com is the best blog, but it has 4,593 posts. The top 10 blogs had more than 21,000 posts in 2006. Under the unit cost model, large blogs are important, but reading a blog with many posts is very time-consuming. This motivates the *number of posts* (NP) cost model, where we set the cost of a blog to the number of posts it had in 2006.

Comparing the NP cost model with the unit cost in Figure 5b, we note that under the unit cost model, CELF chooses expensive blogs with many posts. For example, obtaining the same objective value requires reading 10,710 posts under the unit cost model. The NP cost model achieves the same score while reading just 1,500 posts. Thus, optimizing the benefit:cost ratio leads to drastically improved performance.

Interestingly, the solutions obtained under the NP cost model are very different from the unit cost model. Rather than picking large blogs that capture many stories, under the NP model, the algorithm discovered the notion of *summarizer* blogs (for example, themodulator.org or watcherofweasels.com); these bloggers navigate the blogosphere and discover and talk about stories that eventually become big. Blogs selected under NP cost appear about three days later in the cascade than those selected under unit cost, which further suggests that summarizer blogs tend to be chosen under the NP model.

Although picking good blogs is important, there may not be a single blog, or even a small set of blogs, covering your personal interests. To address this issue, we can reverse our information management goal. Rather than selecting blogs, we can find a principled approach for picking a set of posts that best covers the important stories in the blogosphere. This task can also be formalized as a submodular optimization problem.[10] As an example, here is the set of posts that our algorithm selects out of the 200,000 posts in an eight-hour period on 18 January 2009 if our budget k is set to five:

1. Israel unilaterally halts fire as rockets persist
2. Downed jet lifted from ice-laden Hudson River
3. Israeli-trained Gaza doctor loses three daughters and niece to IDF tank shell
4. EU wary as Russia and Ukraine reach gas deal
5. Obama's first day as president: prayers, war council, economists, White House reception

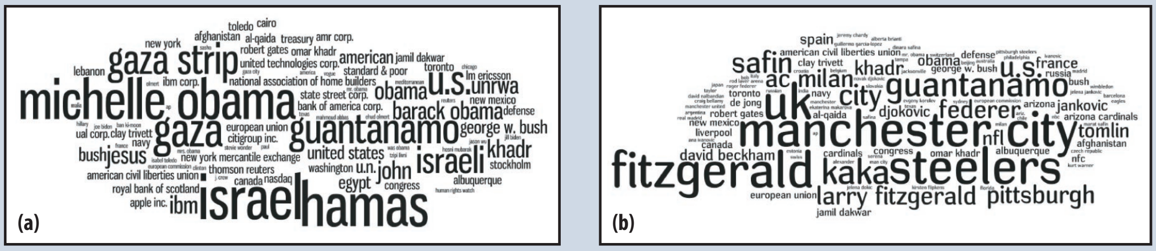

Figure 6. Important words in blog postings on 21 January 2009 (a) before and (b) after personalization. These figures were generated using wordle.net.

The selected five posts all cover the most prevalent stories from this particular day.

But the most prevalent stories may not be the ones that are important to you. Since people have varied interests, the ideal coverage algorithm should incorporate user preferences to tailor the selected posts to individual tastes.

For this task, we must learn a personalized coverage function for the blogosphere through user interaction—for example, from users labeling posts they would like to read or not read, or by what posts the user clicked on.[10] As an example, Figure 6a illustrates the important words being discussed in the blogosphere on 21 January 2009, while Figure 6b shows the important words for the articles the algorithm picks for a user who is very interested in sports. Such a personalization problem exemplifies problems where, rather than *being given* a submodular function, we need to address the problem of *learning* a submodular function from data. Here again we were able to exploit this mathematical structure to develop an algorithm with strong theoretical guarantees that performs very well over these huge datasets.

Our society is drowning in information: The Internet allows us to access virtually every bit of information available to humanity at any given time and at virtually no cost. The ubiquitous availability of sensors allows monitoring of the physical world in an unprecedented manner. This development presents exciting new opportunities for the advancement of science and society. However, this explosion in availability of information also creates the fundamental challenge of developing new techniques for extracting the most useful information. We believe that the algorithms we describe present an interesting step in this direction. ◼

References

1. A. Singh et al., "Efficient Informative Sensing Using Multiple Robots," *J. Artificial Intelligence Research*, Apr. 2009, pp. 707-755.
2. A. Ostfeld et al., "The Battle of the Water Sensor Networks (BWSN): A Design Challenge for Engineers and Algorithms," submitted to *J. Water Resources Planning and Management*, 2008.
3. A. Krause et al., "Efficient Sensor Placement Optimization for Securing Large Water Distribution Networks," *J. Water Resources Planning and Management*, vol. 136, no. 6, 2008, pp. 516-526.
4. G. Nemhauser, L. Wolsey, and M. Fisher, "An Analysis of the Approximations for Maximizing Submodular Set Functions," *Mathematical Programming*, vol. 14, 1978, pp. 265-294.
5. L.A. Rossman, "The EPANET Programmer's Toolkit for Analysis of Water Distribution Systems," *Proc. Ann. Water Resources Planning and Management Conf.*; http://cedb.asce.org/cgi/WWWdisplay.cgi?9903165.
6. A. Krause et al., "Robust Submodular Observation Selection," *J. Machine Learning Research*, Dec. 2008, pp. 2761-2801.
7. A. Das and D. Kempe, "Algorithms for Subset Selection in Linear Regression," *Ann. ACM Symp. Theory of Computing*, ACM Press, 2008, pp. 45-54.
8. M. Kinsley, "How Many Blogs Does the World Need?" *TIME Magazine*, Nov. 2008; www.time.com/time/magazine/article/0,9171,1860888,00.html.
9. J. Leskovec et al., "Cost-Effective Outbreak Detection in Networks," *Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining* (KDD 07), ACM Press, 2007, pp. 420-429.
10. K. El-Arini et al., "Turning Down the Noise in the Blogosphere," *Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining* (KDD 09), ACM Press, 2009, pp 289-298.

Andreas Krause *is an assistant professor of computer science in the Division of Engineering and Applied Science at the California Institute of Technology. His research interests are in adaptive systems that actively acquire information, reason, and make decisions in large, distributed, and uncertain domains, such as sensor networks and the Web. He received a PhD in computer science from Carnegie Mellon University. He is a member of the IEEE, the ACM, and AAAI. Contact him at krausea@caltech.edu.*

Carlos Guestrin *is the Finmeccanica Associate Professor in the Machine Learning and Computer Science Departments at Carnegie Mellon University. His research interests are learning and inference in probabilistic models, machine learning over parallel and distributed computer architectures, and applying these methods to the challenges of sensor networks and information overload. He received a PhD in computer science from Stanford University. He is a member of AAAI and the ACM. Contact him at guestrin@cs.cmu.edu.*