Worst Case Analysis : Our Strength is Our Weakness

Michael Mitzenmacher

Harvard University

Prelude

- Worst-case analysis is dominant in TCS for several reasons.
  - Very suitable for mathematical analysis.
  - Often gives the right, or at least a suitable, answer in practice.
  - Distinguishes us from other fields.
- But it's also limiting, to us and the field.
  - Not all problems best described this way.
    - Worst-case is not the "real-world" case.
  - Pushes us to focus on variations of worst-case analysis.
    - Competitive ratio.
    - Instance optimality.
    - Resource augmentation.
    - Fixed parameter tractability.
    - Others….

Rexford Article

My Ten Favorite "Practical Theory" Papers

Jennifer Rexford
Princeton University
jrex@cs.princeton.edu

Abstract

ABSTRACT

As the saying goes, "In theory there is no difference between theory and practice. But, in practice, there is." Networking research has a wealth of good papers on both sides of the theory-practice divide. However, many practical papers stop short of having a sharp problem formulation or a rigorously considered solution, and many theory papers overlook or assume away some key aspect of the system they intend to model. Still, every so often, a paper comes along that nails a practical question with just the right bit of theory. When that happens, it's a thing of beauty. These are my ten favorite examples. In some cases, I mention survey papers that cover an entire body of work, or a journal paper that presents a more mature overview of one or more conference papers, rather than single out an individual research result. (As an aside, I think good survey papers are a wonderful contribution to the community, and wish more people invested the considerable time and energy required to write them.)

Rexford's Top Ten Papers

- Griffin, Shepherd, Wilgong; The stable paths problem and interdomain routing, IEEE/ACM Trans. on Networking, 2002.
- Chiang, Low, Calderbank, Doyle; Layering as optimization decomposition: a mathematical theory of network architectures, Proc. of the IEEE, 2007.
- Duffield, Grossglauser; Trajectory sampling for direct traffic observation, IEEE/ACM Trans. on Networking, 2001.
- Zhang, Roughan, Duffield, Greenberg; Fast accurate computation of large-scale IP traffic matrices from link loads, ACM Sigmetrics, 2003.
- Estan, Savage, Varghese; Automatically inferring patterns of resource consumption in network traffic, Proc. ACM SIGCOMM, 2003.

Rexford's Top Ten Papers

- Broder, Mitzenmacher; Network applications of Bloom filters: A survey, Internet Mathematics, 2004.
- Parekh, Gallager; A generalized processor-sharing approach to flow control in integrated services networks: The single node case, IEEE/ACM Trans. on Networking, 1993.
- Salehi, Zhang, Kurose, Towsley; Supporting stored video: Reducing rate variability and end-to-end requirements through optimal smoothing, IEEE/ACM Trans. on Networking, 1998.
- Harchol-Balter, Downey; Exploiting process lifetime distributions for dynamic load balancing, ACM Transactions on Computer Systems, 1997.
- Mitzenmacher, Richa, Sitaraman; The power of two random choices: A survey of techniques and results. Handbook of Randomized Computing.

Lessons

- I am shameless.

Lessons

- I am shameless.
- Rexford has impeccable taste.

Lessons

- I am shameless.
- Rexford has impeccable taste.
- *Theory is missing something.*

Lessons

- I am shameless.
- Rexford has impeccable taste.
- *Theory is missing something.*
  - I'm not saying I agree with these choices, but it's a rather disappointing outside-of-theory viewpoint.

Lessons

- I am shameless.
- Rexford has impeccable taste.
- *Theory is missing something.*
  - I'm not saying I agree with these choices, but it's a rather disappointing outside-of-theory viewpoint.
  - A common theme in many of these papers is that they aren't focused on worst-case analysis, but coming up with a good solution to a real problem.

Recent Theory Misses

- Areas where theory has come a little too late to the party.
  - Belief propagation.
  - Cavity method/survey propagation.
  - Network coding.
  - Compressed sensing.
  - Mean field analysis/fluid models.
  - Polar codes.
  - Most any heuristic method (ant colony optimization, tabu search, etc.)

My Two Points

- Theory should better promote an environment where work on real-world problems and issues is appreciated, even at the expense of complete formal analysis.
- Theory should continue to push to expand the analytic tools and methods we have available for non-worst-case analysis.

My Two Points

- Theory should better promote an environment where work on real-world problems and issues is appreciated, even at the expense of complete analysis.
  - Example : work on heuristics.
- Theory should continue to push to expand the analytic tools and methods we have available for non-worst-case analysis.
  - Example : more understanding of “random input” analysis.

Putting My Money Where My Mouth Is

- Heuristics
- Hashing + Entropy

Heuristics

- I did some work in heuristics.
  - Human-Guided Tabu Search (AAAI 2002)
  - A Complete and Effective Move Set for Simplified Protein Folding (RECOMB 2003)
  - New Heuristic and Interactive Approaches to 2D Rectangular Strip Packing (JEA 2005)
  - BubbleSearch (IPL 2006)
- Entered a skeptic.
- Left with more respect.
  - The area has more to it than we might think, even if it's less rigorous than we would want.
  - The area could benefit from theory approaches and insight.
  - Lots of real problems that need to be solved.

Human-Guided Tabu Search

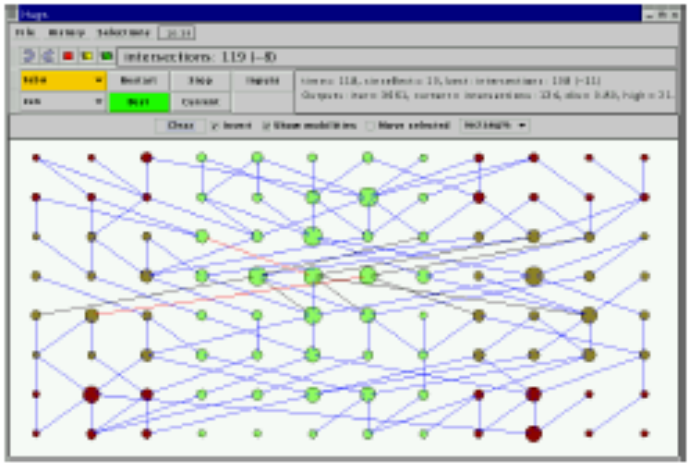


Figure 1: The Crossing Application.

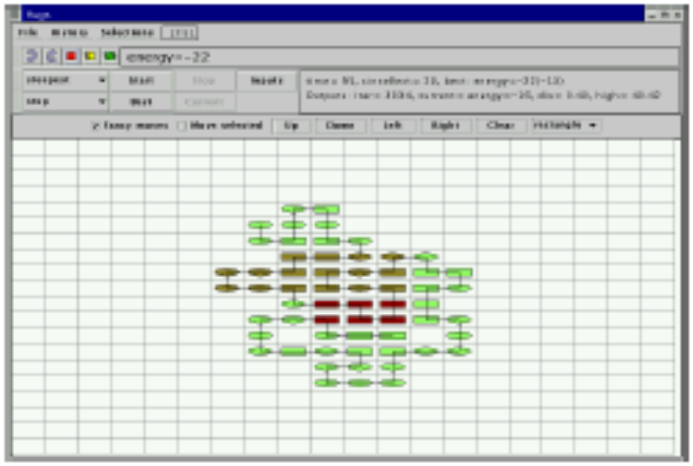


Figure 3: The Protein Application.

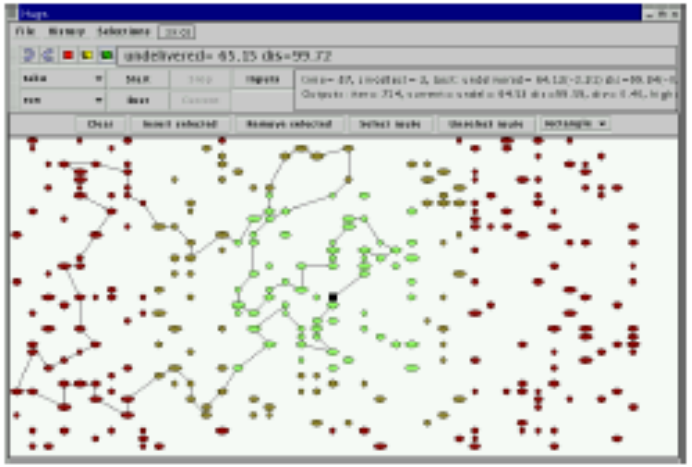


Figure 2: The Delivery Application.

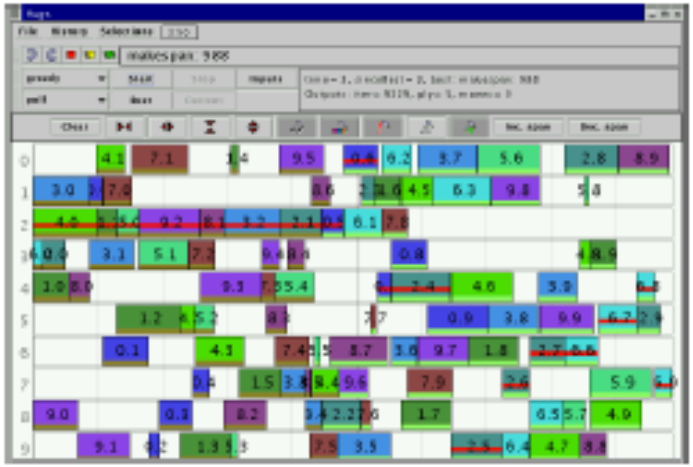
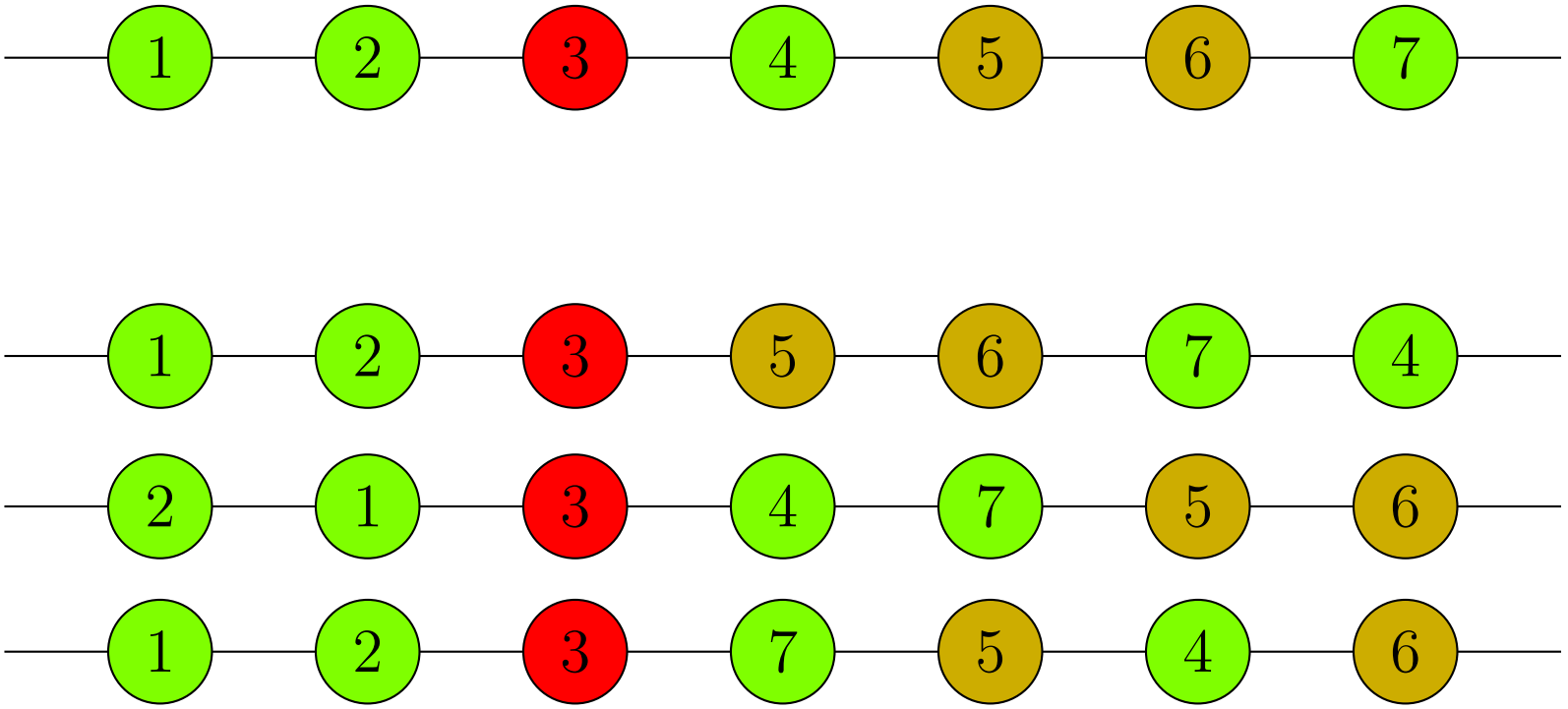


Figure 4: The Jobshop Application.

Stoplight Framework



Greedy Algorithms

- Many standard Greedy algorithms have two steps:
  - Order elements
  - Place elements sequentially
    - Given a partial solution of k elements and a next element, place the next element.
    - Often ordering is dynamically adjusted after a placement : fixed priority vs. dynamic priority.
- Examples
  - Bin-packing : Best Fit Decreasing, First Fit Decreasing
  - Set cover/vertex cover
  - JobShop scheduling : schedule subjobs by work remaining, place each subjob as early as possible.
  - Many others…

Theorist Viewpoint

- Most Greedy algorithms are not optimal.
  - But they appear to work well as heuristics.
- Let's apply worst-case analysis.
  - Competitive ratio: how far from the optimal can you be?
- Competitive ratio becomes the new metric.
- Theory Problem: Let's find algorithms with better competitive ratios.
- Original Problem: Let's get better solutions.
- Not clear the theory abstraction is helping us with the original problem.

Randomized Greedy

- The problem with Greedy algorithms is that they present 1 solution.
  - If time is available, should look for better solutions.
- Randomized Greedy algorithms: keep trying "near-greedy" solutions until time runs out.
- TOP-k: choose next element placed uniformly from top k elements.

BubbleSearch

- BubbleSearch: go through the elements in order sequentially, flipping a coin with probability p for heads for each element. At first heads, place that element (and go back to start).
- Probability of choosing an ordering with BubbleSort distance (= Kendall-tau distance) D from the original Greedy ordering is proportional to

$$(1 - p)^D$$

Other BubbleSearch properties

- Anytime algorithm: runs until you stop (returns best solution so far).
- All element orderings are possible.
- Follows the intuition behind the Greedy ordering: better to place top things in the orders first, but more flexibly.
- More robust, flexible than TOP-k approach.
- Useful variant : with fixed priorities, change the “base ordering” when a new best solution is found.

Effectiveness

- BubbleSearch seems effective whenever it is used.
  - Usually finds a better-than-greedy solution quickly – a few hundred iterations.
  - Margin over Greedy depends on how good Greedy is alone.
- It's brain-dead simple.
  - You should be teaching this in your undergraduate algorithms class. (I do.)
  - If you've implemented a Greedy algorithm, this additional layer is easy to implement.
    - Our implementation required only hooks into ordering, placement functions.
  - vs. other possible heuristics : e.g., tabu search, etc.

Aside

- Theorists think in terms of computational resources.
  - Time, space are the obvious ones.
  - I teach (in undergraduate algorithms) that "correctness" is a resource, can be traded off with time and space.
- Programmer time is the resource theorists seem to think least about, arguably often the most important.
  - Simple, general solutions are good.

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Processing Letters 97 (2006) 161–169
Information Processing Letters
www.elsevier.com/locate/ipl

BubbleSearch: A simple heuristic for improving priority-based greedy algorithms

N. Lesh [a], M. Mitzenmacher [b,1,*]

Abstract

We introduce BubbleSearch, a general approach for extending priority-based greedy heuristics. Following the framework recently developed by Borodin et al., we consider *priority algorithms*, which sequentially assign values to elements in some fixed or adaptively determined order. BubbleSearch extends priority algorithms by selectively considering additional orders near an initial good ordering. While many notions of nearness are possible, we explore algorithms based on the Kendall-tau distance (also known as the BubbleSort distance) between permutations. Our contribution is to elucidate the BubbleSearch paradigm and experimentally demonstrate its effectiveness.

Theory of BubbleSearch

- I don’t have one.
- But related to Priority Algorithms framework of Allan Borodin and others.
- Claim : Useful even if not able to theoretically formalize why.
- I think there’s more room for theory CS to offer useful ideas even if they can’t prove things about them.

Heuristics

- There's a whole field of metaheuristics.
- Hard to prove rigorous results.
  - Theorists have tried, in some cases.
- Can theory still give guidance to the area?
  - Coming up with good, new heuristic methods.
    - Based on theory, or not.
  - Improving, judging, providing understanding for existing methods.
  - Devising explanations for why methods often work well (better than our worst-case algorithms).
    - Even if full of somewhat unrealistic assumptions.

Randomness

- CS theory starts with the worst-case inputs, opens up to random cases.
  - Algorithms on random graphs (or other random/semi-random models).
  - Smoothed analysis.
- Some other fields seem to go the other way.
  - Coding theory : binary symmetric channel.
  - Queueing theory : M/M/1 queues (exponential service/ Poisson arrivals)
  - Stochastic control theory
  - Stochastic programming
  - Statistical physics/mechanics

Randomness vs. Worst-Case

- Many settings are not modeled well by worst-case.
  - Though of course some are.
- Most settings are not modeled well by uniformly at random.
  - We can adjust: work on $G_{n,p}$ random graph models has been replaced by work on power-law-graphs of various types.
- Need better understandings of reasonable random and "semi-random" models.
  - A direction we seem to be moving in naturally.

One Approach: Entropy and Hashing

- From paper "Why Do Simple Hash Functions Work?" by Mitzenmacher/Vadhan.
  - Simple = chosen from a pairwise (or *k*-wise) independent (or universal) family.
    - Our results are actually more general.
  - Work = perform just like random hash functions in most real-world experiments.
- Motivation: Close the divide between theory and practice.

Universal Hash Families

- Defined by Carter/Wegman
- Family of hash functions Λ of form $H:[N] \rightarrow [M]$ is k-wise independent if when H is chosen randomly, for any $x_1, x_2, \ldots x_k$, and any $a_1, a_2, \ldots a_k$,

$$\prod_{i=1}^{k} \Pr(H(x_i) = a_i) = 1/M^k$$

- Family is k-wise universal if

$$\Pr(H(x_1) = H(x_2) \ldots = H(x_k)) \leq 1/M^{k-1}$$

Not Really a New Question

- "The Power of Two Choices" = "Balanced Allocations." Pairwise independent hash functions match theory for random hash functions on real data.
- Bloom filters. Noted in 1980's that pairwise independent hash functions match theory for random hash functions on real data.
- But *analysis* depends on perfectly random hash functions.
  - Or sophisticated, highly non-trivial hash functions.

Practical Performance Of Bloom Filters and Parallel Free-Text Searching

M. V. Ramakrishna

The results and the details of the experiments presented in the next section illustrate that transformations chosen at random from the class H_1 yield the theoretical performance of the Bloom filters. (The class H_1 is a *universal*$_2$ class of hashing functions. Definitions and theoretical investigations of *universal*$_2$ classes of hashing functions may be found in [2].)

Worst Case : Simple Hash Functions Don't Work!

- Lower bounds show result cannot hold for "worst case" input.
- There exist pairwise independent hash families, inputs for which Linear Probing performance is worse than random [PPR 07].
- There exist *k*-wise independent hash families, inputs for which Bloom filter performance is provably worse than random.
- Open for other problems.
- Worst case does not match practice.

Random Data?

- Analysis usually trivial if data is independently, uniformly chosen over large universe.
  - Then all hashes appear “perfectly random”.
- Not a good model for real data.
- Need intermediate model between worst-case, average case.

A Model for Data

- Based on models of semi-random sources.
  - [SV 84], [CG 85]
- Data is a finite stream, modeled by a sequence of random variables $X_1, X_2, \ldots X_T$.
- Range of each variable is $[N]$.
- Each stream element has some *entropy*, conditioned on values of previous elements.
  - Correlations possible.
  - But each element has some unpredictability, even given the past.

Intuition

- If each element has entropy, then extract the entropy to hash each element to near-uniform location.
- Extractors should provide near-uniform behavior.

Notions of Entropy

- max probability : $\mathrm{mp}(X) = \max_x \Pr[X = x]$
  - min-entropy : $\mathrm{H}_\infty(X) = \log(1/\mathrm{mp}(X))$
  - block source with max probability *p* per block

$$\mathrm{mp}(X_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) \leq p$$

- collision probability : $\mathrm{cp}(X) = \sum_x (\Pr(X = x))^2$
  - Renyi entropy : $\mathrm{H}_2(X) = \log(1/\mathrm{cp}(X))$
  - block source with coll probability *p* per block

$$\mathrm{cp}(X_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) \leq p$$

- These "entropies" within a factor of 2.
- We use collision probability/Renyi entropy.

Leftover Hash Lemma

- A “classical” result (from 1989).
- Intuitive statement: If $H : [N] \rightarrow [M]$ is chosen from a pairwise independent hash function, and X is a random variable with small collision probability, $H(X)$ will be close to uniform.

Leftover Hash Lemma

- Specific statements for current setting.
  - For 2-universal hash families.
- Let $H:[N]\to[M]$ be a random hash function from a 2-universal hash family Λ. If cp(X)< $1/K$, then (H,H(X)) is $(1/2)\sqrt{M/K}$ -close to (H,$U_{[M]}$).
  - Equivalently, if X has Renyi entropy at least log M + 2log(1/ε), then (H,H(X)) is ε-close to uniform.
- Let $H:[N]\to[M]$ be a random hash function from a 2-universal hash family. Given a block-source with coll prob $1/K$ per block, (H,$H(X_1)$,.. $H(X_T)$) is $(T/2)\sqrt{M/K}$-close to (H,$U_{[M]^T}$).
  - Equivalently, if X has Renyi entropy at least log M + 2log(T/ε), then (H,$H(X_1)$,.. $H(X_T)$) is ε-close to uniform.

That's About It

- We did more in the paper.
- But the main message is:

A weak hash function, on "sufficiently random" data with enough entropy, will behave like a random hash function, with each data item appearing to be hashed randomly, because even weak hash functions are randomness extractors.

Applications

- Potentially, wherever hashing is used
  - Bloom Filters
  - Power of Two Choices
  - Linear Probing
  - Cuckoo Hashing
  - Many Others…

Related Work

- Vadhan and Chung improve the bounds.
- Dietzfelbinger and Schellbach warn: let the buyer beware! Weak hash functions fail in some standard settings.
  - Small universe of key-set, so not enough entropy.
- Pandurangan and Upfal investigate entropy-based bounds for online algorithms.
  - Works for some, not others.
- Patrascu and Thorup on tabular hashing.
  - Their results are worst-case, but just give a "new view" on tabular hashing.

Open Questions

- Are these ideas more generally useful?
  - Is the approach particular to hashing, as it depends on hash functions being extractors?
  - Other possible uses?
- More generally, can we use "weaker" notions of randomness in the analysis of algorithms?
  - What notions are suitably more realistic?

Conclusion 1

- The Field's Mindset : Theoretical Computer Science (FOCS/STOC/SODA) is about Proofs.
  - It's what separates us from the unwashed masses.
- Can we loosen that self-conception a bit?
  - Algorithms is about solving problems.
  - Sometimes, we may not have a proof.
  - That doesn't mean there's not "theory" there.
    - Is good design, based on theory, theory?

Conclusion 2

- We need more expansive models/techniques.
- A lot of our "different models" are worst-case models with (suitable) add-ons.
- I like methods and models based on randomness.
  - TCS has a rich history there, and still expanding.
  - But we're nowhere close to done.
- I think we need our models to be more closely tied to reality.
  - We need to work more in the trenches – "solve" real problems – to do so.
  - Or we may be considered followers, not leaders.
- I think we need to be bolder in allowing more assumptions, to understand why algorithms work "surprisingly well" on most inputs, while not working well on all inputs.