15-251
Great Theoretical Ideas in Computer Science

Announcements

Final Exam: Tuesday Dec 14, 8:30-11:30am
In GHC 4401 (Rashid)

Review Session: Err…Dunno? This will be posted.

Plan for Today:

1: On-Line Algorithms

2: Look at Funny Pictures

3: Voting Theory

4: Voting a topic off the final

5: Cupcake eating contest

Online Algorithms

Subtitle: A success of Theory

Lecture 29 (Dec 2, 2010)

"Online" algorithms

NP-hardness is not the only hurdle
we face in day-to-day algorithm design

Lack of information is another…

E.g. Scheduling Jobs on Machines

Input:
A set of n jobs, each job j has processing time p_j
A set of m identical machines

Online Algorithms

Instead of the jobs being given up-front

they arrive one by one (in adversarial order)

you have to schedule each job before seeing the next one.

What's a good algorithm?

Graham's Greedy Algorithm!

Graham's Greedy Algorithm

Order the jobs $j_1, j_2, \ldots, j_n$ in some order

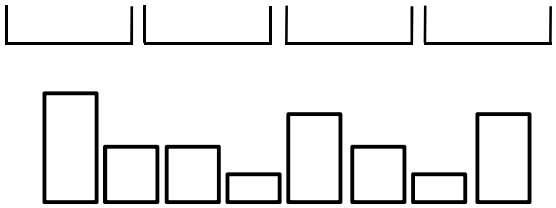
Initially all the machines are empty

For t = 1 to n

Assign j_t to the least loaded machine so far

In fact, this "online" algorithm performs within a factor of (2-1/m) of the best you could do "offline"

Moreover, you did not even need to know the processing times of the jobs when they arived.



Online Algorithms

These algorithms see the input requests one by one, and have to combat lack of information from not seeing the entire sequence.

(and maybe have to combat NP-hardness as well).

Example: List Update

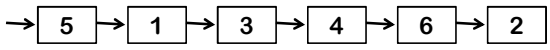
You have a linked list of length n

Each item in this list is a (key, data) pair

Given a key as a request, you traverse the list until you get to the relevant item. You pay 1 for each link you traverse.

You are now allowed move the requested item up the list for free.

Given a sequence of key requests online, what should you do?



Ideal Theorem (for this lecture)

The cost incurred by our algorithm on any sequence of requests

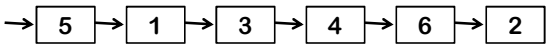
is "not much more" than

the cost incurred by any algorithm on the same request sequence.

(We say our algorithm is "competitive" against all other algorithms on all request sequences)

Does there exist a "competitive" algorithm?

let's see some candidates…

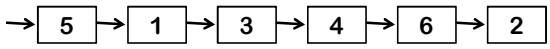


Algorithm "Do nothing":

Request sequence: 2,2,2,2,2,2,2,2,…
incurs cost 6 for each request

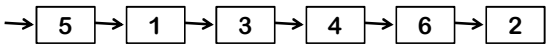
Does badly against the algorithm which first moves 2 to the head of the list (paying 6), and pays 1 from then on.

not competitive ☹



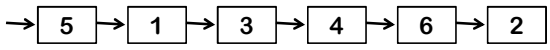
Algorithm "Transpose":

Every time it sees an element, moves it one place closer to the head of the list



Algorithm "Frequency Counter":

Maintain the list in sorted order of the access frequencies



Algorithm "Move to Front":

Every time it sees an element, moves it up all the way to the head of the list

Which is best?

Algorithm "Transpose":

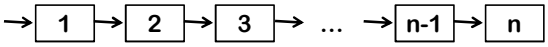
Algorithm "Move to Front":

Algorithm "Frequency Counter":

For both Transpose and Freq-Count
there are request sequences where
they do far worse than other algorithms

they are not competitive ☹

Bad example for Transpose

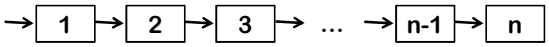


Request sequence =
n,n-1,n,n-1,n,n-1,…

Transpose incurs cost n each time

Best strategy: move them to front (pay 2n),
now cost 1 each time.

Bad example for Freq-Count



Request sequence =
1 (n times), 2 (n times), …, n (n times), …

Freq-Count does not alter list structure
$\Rightarrow$ pays $n\Delta_n = \Theta(n^3)$ on each set of n requests

Good strategy:
Moves the item to the front on first access,
incurs cost $\Delta_n + n^2$

Algorithm: Move-to-Front

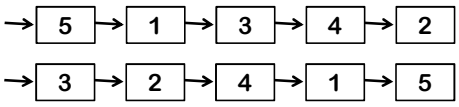
Theorem: The cost incurred by Move-to-Front
is at most twice the cost incurred by <u>any</u> algorithm

even one that knows the entire request
sequence up-front!

[Sleator Tarjan '85]

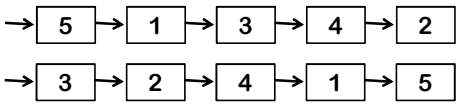
MTF is "2-competitive" ☺

Proof is simple but clever. Uses the idea
of a "potential function" (cf. 15-451)



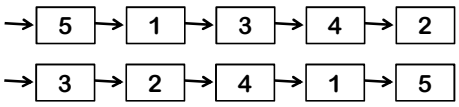
Observation: if our list and the other algo's
lists are in the same order, we incur same cost.

Natural thing to keep track of:
number of pairs on which we disagree.



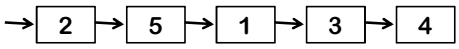
Number of transposed pairs =

12 13 14 15
23 24 25
34 35
45



Number of transposed pairs =

Suppose we get request 2 (cost for us = 4, them = 1)



But number of transposed pairs decreased by 2

(25, 21, 24 corrected, 23 broken)

Theorem: MTF is 2-competitive

Theorem: No (deterministic) algorithm can do better

Theorem: Random MTF (a.k.a. BIT) is better!

Each key also stores an extra bit.
Initialize bits randomly to 0 or 1.

When key is requested, flip its bit.

When key's bit is flipped to 1, move to front.

This is a 1.75-competitive algorithm!

Many cool combinatorial problems that arise in analyzing algorithms.

The mathematical tools you've learnt in this course are extremely relevant for their analysis.

Plan for Today:

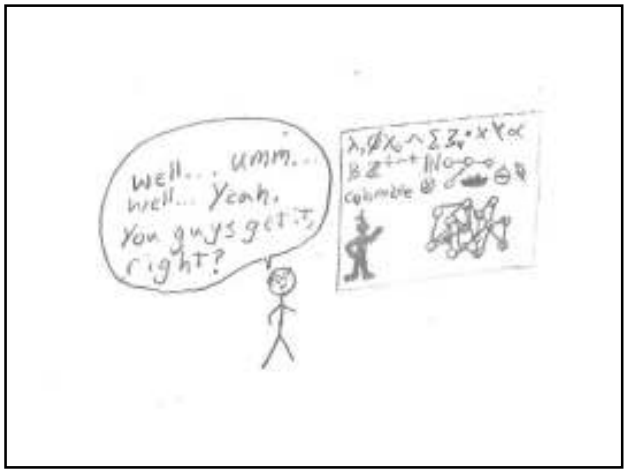
1: On-Line Algorithms

2: Look at Funny Pictures

3: Voting Theory

4: Voting a topic off the final

5: Cupcake eating contest



How Should We Vote?

Subtitle: A failure of Theory

Lecture 29 (Dec 2, 2010)

n.b.:
Danny's views,
not necessarily
mine
--Anupam

Part 1: The System is Broken

Proof: The 2000 election.

Presidential candidate	Vote total	%	Party
George W. Bush (W)	2,912,790	48.847	Republican
Al Gore	2,912,253	48.838	Democratic
Ralph Nader	97,421	1.634	Green
Patrick J. Buchanan	17,484	0.293	Reform
Harry Browne	16,415	0.275	Libertarian
John Hagelin	2,281	0.038	Natural Law/Reform
Howard Phillips	1,378	0.023	Constitution
Other	3,025	0.051	—
Total	**5,963,110**		

Source: *2000 official presidential general election results*

QED. (There are many other examples)

The system we use (called plurality voting, where each voter selects one candidate) doesn't work well for 3 or more candidates.

Clearly the "wrong" candidate often wins.

By "wrong" I mean there is a losing candidate who would make more people happier than the winner. (We'll get to defining this more precisely later.)

Little known tangential fact:

Actually, Gore won the election, as shown in a full statewide recount down by a consortium of newspapers.

http://www.nytimes.com/2001/11/12/politics/recount/12ASSE.html

Part 2: Ranked Ballots

Nicolas de Caritat, marquis de Condorcet, 1743 to 1794

He studed the concept of ranked ballots – having the voters rank all the candidates

Concorcet's Analysis

For each pair of candidates, decide who is preferable. (i.e. wins in more of the rank orderings)

In these matchups, if there's one candidate who beats all, he/she is the clear winner.

This candidate is called the Condorcet Winner

Example. Three candidates B, G, and N.

1000	B > G > N
500	G > B > N
500	G > N > B
10	N > G > B
1	N > B > G

Total of 2111 votes.

B>G 1001 G>B 1010
B>N 1500 N>B 511
G>N 2000 N>G 11

G is the Condorcet winner

Concorcet's Paradox

1	A > B > C
1	B > C > A
1	C > A > B

So we have A>B, B>C and C>A

There might not be a Condorcet winner.

Proposed Solutions

Dozens of solutions have been proposed.

Two of them are:

Borda Counting
Instant Runnof Voting (IRV)

Borda Counting

There are n candidates.

Assign a score by each voter to each candidate. n to the best, n-1 to the next and so on down to 1 for the least.

Now compute the candidate with the highest total.

Instant Runoff Voting (IRV)

There are n candidates.

Repeat until there's just one candidate left:

> Find the candidate with the least #1 rankings.
>
> Delete that candidate from all ballots.

Borda and IRV are better than plurality, but is there a really good system?

The answer is "NO". Kenneth Arrow proved in 1950 that Democracy is impossible.

Things are hopeless. Forget about it.

Ok, calm down. What did he actually prove?

Say you have an election function F that takes as input the rank orderings of all the voters and outputs a rank ordering.

F(v1, v2, v3,…,vn)

(F is deterministic and not necessarily symmetrical on its inputs.)

It would be nice if F had the following properties:

1. (U) Unanimity If all votes have A>B then the output has A>B.
2. (IIA) Independence of irrelevant alternatives: If we delete a candidate from the election, then the outcome is the same except with that candidate missing.

Arrow's Theorem:

Any voting function that handles 3 or more candidates and satisfies U and IIA is a dictatorship!

(A dictatorship I mean that there's one voter who dictates the entire outcome of the election.)

The proof is not too difficult.

Arrow won the Nobel Prize in economics in part for this work.

This theorem derailed the entire field of social choice theory for the last 50 years, as we'll see.

Wait, you say.

We really only want to determine a winner. We don't need the election function to generate a full rank ordering. Surely we can do that.

Good point. But you're out of luck there too.

In the 1970s Gibbard and Satterthwaite proved this: There does not exist a winner selection algorithm satisfying these properties:

1. The system is not a dictatorship
2. If every voter ranks A on top, then A wins
3. It's deterministic
4. There are at least three candidates
5. It never pays for voters to lie. That is, if a voter V prefers A to B, then putting B before A in her vote cannot cause a better outcome from her point of view.

Part 3: Range Voting

What about the kind of voting we use all the time on the internet. Like at Amazon.com, or HotOrNot, or MRQE?

Every voter scores each candidate on a scale of, say 1 to 10. Then order the candidates by their average vote.

The idea is called range voting (aka score voting).

Let's think about the criteria listed in Arrow's theorem.

Does range voting satisfy unanimity?

Of course. If each voter scores A above B then A will have a higher average than B

Does range voting satisfy IIA?

Of course. If we delete one or more candidates from the election, then the rest stay the same.

Is range voting a dictatorship?

No. Duh.

RANGE VOTING DOES THE IMPOSSIBLE!

How does it do that?

We've changed the rules of the game laid out by Concordet, and followed by the entire field of social choice for 250 years.

We don't restrict voting to preference lists. We allow scores. This tiny change fixes these problems.

Oh, and what about the Gibbard Satterthwaite theorem?

Again, range voting does the "impossible".

It satisfies all the criteria at least for three person elections.

See: http://www.rangevoting.org/GibbSat.html

But is there a better way to analyze voting systems?

Enter Warren Smith in the late 1990s.

Smith applied a system called Bayesian Regret to the analysis of voting systems.

Oddly, this had never been applied to voting systems before.

Bayesian Regret Simulations

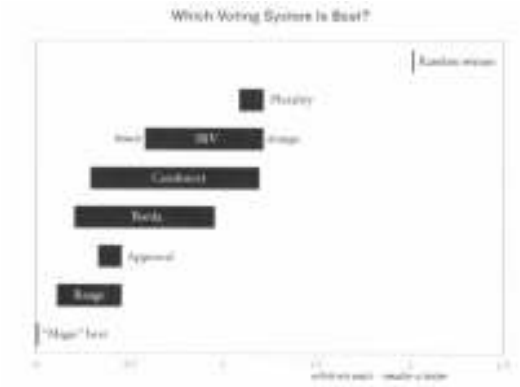
1. Each voter has a personal "utility" value for the election of each candidate
2. Now the voters vote, based both on their private utility values, and (if they are strategic voters) on their perception from "pre-election polls" (also generated artificially within the simulation, e.g. from a random subsample of "people") of how the other voters are going to act.
3. The election system E elects some winning candidate W.
4. The sum over all voters V of their utility for W, is the "achieved societal utility."
5. The sum over all voters V of their utility for X, *maximized* over all candidates X, is the "optimum societal utility" which would have been achieved if the election system had magically chosen the societally best candidate.
6. The difference between 5 and 4 is the "Bayesian Regret" of the election system. It is zero if W=X, or it could be positive if W and X differ.

See http://www.rangevoting.org/BayRegDum.html

Warren Smith's Simulations

Smith simulated millions of election scenarios, adjusting the distribution of strategic voters, and the distributions of private utility values.

Range voting worked the best in *ALL* of the simulations.



The moral of the story

1. In theory we often make assumptions in order to prove theorems.

   Be careful how you interpret and use the theorems. They can be misleading.

   EG: Voting is impossible..

   EG: Don't even bother to try to solve NP complete problems. It's hopeless.

2. Range Voting is the best voting system.

References

www.rangevoting.org

Gaming the Vote -- Why elections aren't fair, and what we can do about it
by William Poundstone, 2008

We Had Some Lectures

1. Pancakes with a Problem
2. Inductive Reasoning
3. Ancient Wisdom: Unary and Binary
4. Solving Problems, Writing Proofs
5. Games I: Which Player Wins?
6. Games II: Nimbers
7. Counting I: Choice Trees…
8. Counting II: Pascal, Binomials…
9. Counting III: Generating Functions
10. Propositional Logic
11. Probability I: Basic Probability
12. Probability II: Great Expectations
13. Number Theory
14. Cryptography and RSA
15. Algebraic Structures: Groups, Rings, and Fields
16. Error Correction Codes
17. Probability III:
    - Infinite sample spaces
    - Random Walks
    - Lagrange Interpolation
18. Graphs I: Trees and Planar Graphs
19. Graphs II: Matchings, Tours…
20. Finite Automata and Languages
21. The Stable Marriage Problem
22. How to Add and Multiply
23. Cantor's Legacy: Infinity and Diagonalization
24. Turing's Legacy: The Limits of Computation
25. Gödel's Legacy: What is a Proof?
26. Efficient Reductions Between Problems
27. Complexity Theory: NP vs P
28. Approximation Algorithms
29. On-Line Algorithms & Voting

Time to Vote!

You will use 0-5 range voting to pick a topic to remove from the final.

Mark your vote next to each topic. (No mark indicates a 0 vote.)

We will remove the one with the most votes.

Plan for Today:

1: On-Line Algorithms

2: Look at Funny Pictures

3: Voting Theory

4: Voting a topic off the final

5: Cupcake eating contest

Contest

You are now eating manually.

First person to finish his or her cupcake gets 1% extra credit on the final!

I understand that eating cupcakes can be a dangerous activity and that, by doing so, I am taking a risk that I may be injured.

I hereby assume all the risk described above, even if Anupam Gupta, his TAs or agents, through negligence or otherwise, otherwise be deemed liable. I hereby release, waive, discharge covenant not to sue Anupam Gupta, his TAs or any agents, participants, sponsoring agencies, sponsors, or others associated with the event, and, if applicable, owners of premises used to conduct the cupcake eating event, from any and all liability arising out of my participation, even if the liability arises out of negligence that may not be foreseeable at this time.

Please don't choke yourself…