CS 4850-Lecture 41 –scribes: Heqing Li & Sujith Vidanapathirana

April 29, 2009

I. Continuation of Arrow Theorem Proof

Arrow Theorem

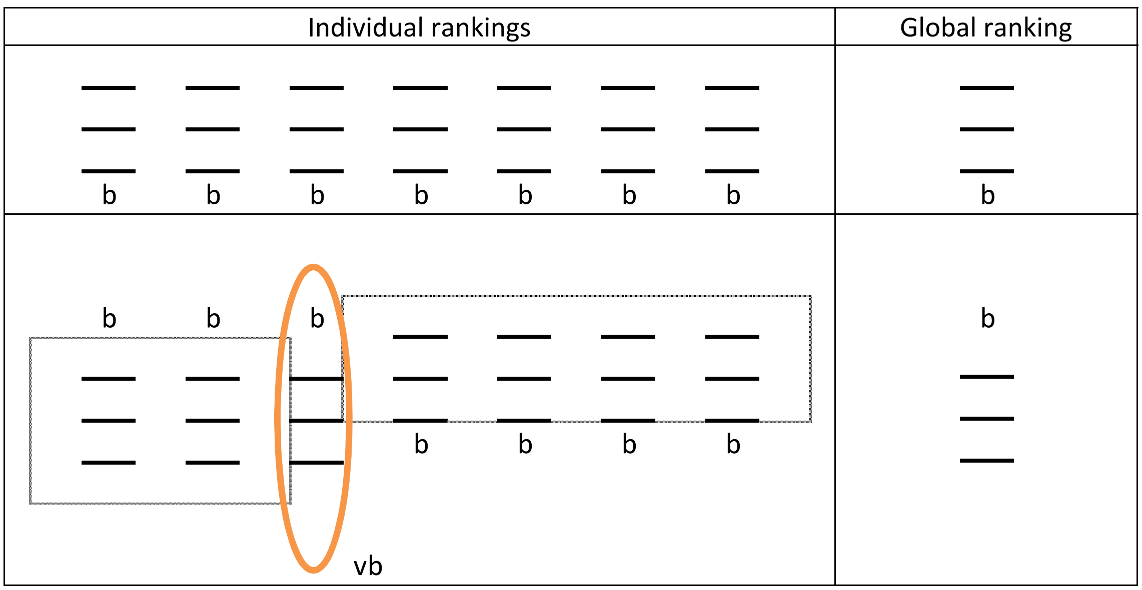
There's no way to satisfy all of the 3 axioms:

1) Unanimity
2) Independency of irrelevant alternatives
3) Non-dictator

We have been trying to prove 1) + 2) => No 3).

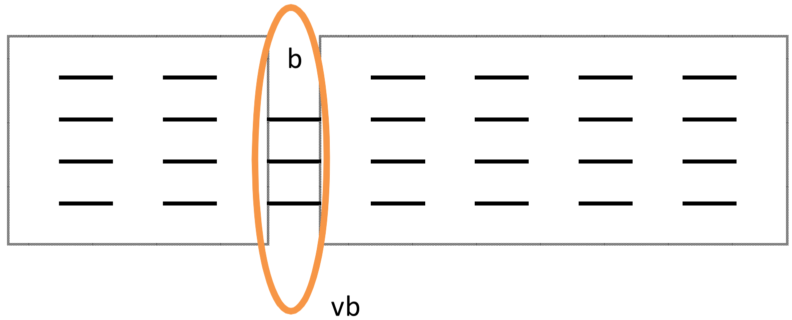
If everyone ranks b as the least preferred, the global ranking would have b at the bottom.

Starting from this preference list, if we move b to the top one by one, somewhere in the process of moving b up to the 1st place, b should become the 1st choice in the global ranking.



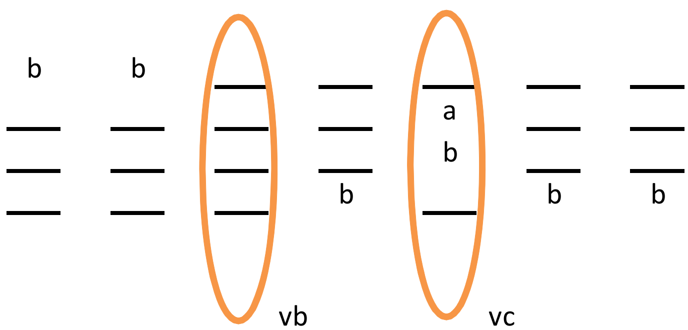
Assume, the non-b ranking entries outside vb are arbitrary. When we move b down in vb, b would move the same way in the global ranking. Vb is a dictator.

For any other candidate being ranked, a, the only time that b can switch order with a in global ranking is when b and a switch in vb. Thus, b must go through the global ranking the same way as in vb.



All the rankings outside vb are arbitrary, vb is almost a dictator, dictator of anything but b.

Then, suppose there's a vc, a dictator for all symbols except c in the case below:



Suppose vb is not the same as vc. If we switch the order of a and b in vb, we would change the order of a and b in global. However, a and b are not switched in vc. This leads to contradiction. Therefore, vc must be the same as vb.

When vb = vc, we have this ranking agrees with the global on anything except b.

a	a
b	c
c	b

If vb has the first, and the global has the second way of relative ranking of the three, if we move b to above a in vb and b to above a in the global, the b-c relative order does not change in vb but changes in the global.

Takeaway:

No way to satisfy all reasonable assumptions. No matter how you do it, you are going to violate a principle.

II. How to store entire World Wide Web on laptop

The goal is to store sufficient data to answer certain questions. For example, how to store data in order to check for plagiarism

A. Simpler problem: storing a set

S is a subset of {1, …, 10^9}, and |S|=1000. How can we compare 10 elements to know whether two sets are the same?

i. Random sampling would not work.

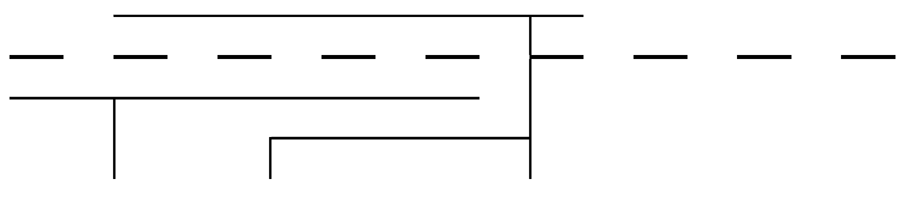
ii. Sort and take 10 min elements would not work. The sets may all include the smallest three or four elements and then random.

iii. Randomized sorting sequence would work.

First choose a randomized sorting sequence, then sort the set according to the sequence. Comparing the smallest 10 would do.

B. Sequence

The key question is how to convert a sequence to a set.



We can combine 5 integers into one, and shift to get the next one.

The new elements can always be reordered.

In this way, the new elements can just form a set. The conversion is solved.

Example: DNA long sequence of 10^9, and short sequence of 10^5.

We can use a linear-time algorithm to match short sequence.

We can use dynamic programming to match sub sequence of the short sequences. Yet this is taking n^3 time, not efficient.

We can use the method described above to do this.

III. Auction

Intuitive rule: The winner of the auction pays his own offer price.

Better rule: The winner of the auction pays the second bidder's offer price.

Rationale: People tend to bid less when the top bidder pays at his own offer price. When the winner only needs to pay the 2^{nd} highest bidder's offer price, he no longer has the incentive to "shave" his offer.

IV. Find frequency of all elements in a sequence with low space requirement

Data stream, S = {a_1, a_2, a_3, a_4, …, a_n} from {1, 2, 3, …, m}. Let H be a set of hash functions that map symbols to {-1, 1}. Randomly select a hash function h in H.

Apply h to S, and we have:

$$s = \sum_{i=1}^{n} h(a_i)$$

This takes at most log n space.

Let n_j be number of occurrences of symbol j, then, we have:

$$\underset{over\ all\ h\ in\ H}{\mathrm{E}}(s \times h(j)) = n_j$$

This is because:

$$\underset{over\ all\ h\ in\ H}{\mathrm{E}}\left(\sum_{i=1}^{n} n_i h(i) \times h(j)\right) = \sum_{i=1}^{n} n_i \underset{over\ all\ h\ in\ H}{\mathrm{E}}(h(i)h(j))$$

$$\underset{over\ all\ h\ in\ H}{\mathrm{E}}(h(i)h(j)) = 1\ if\ i = j$$

$$\underset{over\ all\ h\ in\ H}{\mathrm{E}}(h(i)h(j)) = 0\ if\ i \neq j$$

Thus, we can use this method to find the frequencies, n_j.

The standard deviation makes the estimate error about the real value. Thus, we can do the estimate multiple times and get the error smaller. 10 times, then 10%.