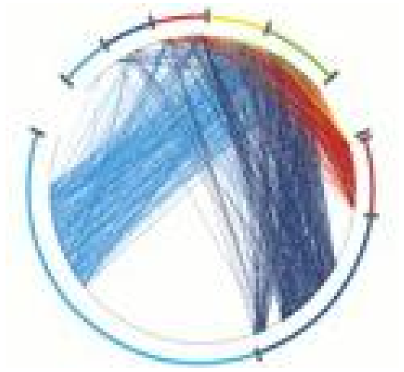
Models and Algorithms for Complex Networks

Theory and Algorithms for Link Analysis Ranking, Rank Aggregation, and Voting



Outline

- Axiomatic Characterizations of Link Analysis Ranking Algorithms
  - InDegree algorithm
  - PageRank algorithm
- Rank Aggregation
  - Computing aggregate scores
  - Computing aggregate rankings - voting

Comparing LAR vectors

🟥 🟦 🟨 🟩 🟪

$$w_1 = [\ 1 \quad 0.8 \quad 0.5 \quad 0.3 \quad 0\ ]$$

$$w_2 = [\ 0.9 \quad 1 \quad 0.7 \quad 0.6 \quad 0.8\ ]$$

- How close are the LAR vectors w_1, w_2?

Distance between LAR vectors

- Geometric distance: how close are the numerical weights of vectors w_1, w_2?

$$d_1(w_1, w_2) = \sum |w_1[i] - w_2[i]|$$

🟥 🟦 🟨 🟩 🟪

w_1 = [1.0 0.8 0.5 0.3 0.0]

w_2 = [0.9 1.0 0.7 0.6 0.8]

$d_1(w_1,w_2)$ = 0.1+0.2+0.2+0.3+0.8 = 1.6

Distance between LAR vectors

- Rank distance: how close are the ordinal rankings induced by the vectors w_1, w_2?
  - Kendal's τ distance

$$d_r(w_1, w_2) = \frac{\text{pairs ranked in a different order}}{\text{total number of distinct pairs}}$$

Similarity

- Definition: Two algorithms A_1, A_2 are similar if

$$\lim_{n\to\infty} \frac{\max_{G\in G_n} d_1\left(A_1(G), A_2(G)\right)}{\max_{w_1, w_2} d_1\left(w_1, w_2\right)} = 0$$

- Definition: Two algorithms A_1, A_2 are rank similar if

$$\lim_{n\to\infty} \max_{G\in G_n} d_r\left(A_1(G), A_2(G)\right) = 0$$

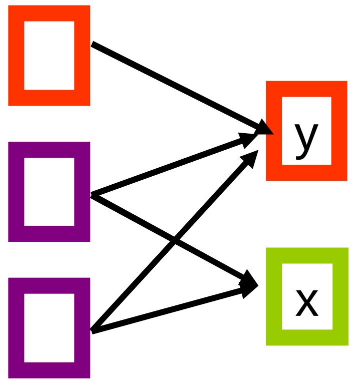
- Definition: Two algorithms A_1, A_2 are rank equivalent if

$$\max_{G\in G_n} d_r\left(A_1(G), A_2(G)\right) = 0$$

Monotonicity

- Monotonicity: Algorithm A is strictly monotone if for any nodes x and y

$$B_N(x) \subset B_N(y) \Leftrightarrow A(G)[x] < A(G)[y]$$



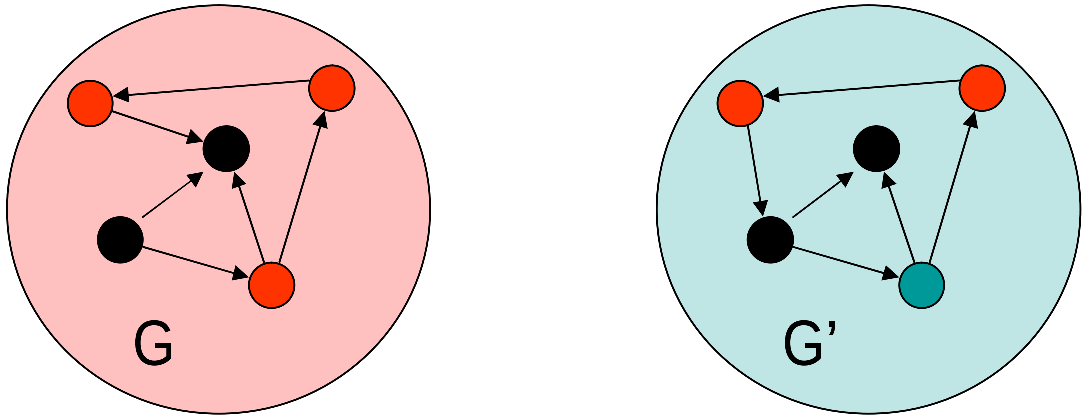
$w_x < w_y$

Locality

- Locality: An algorithm A is strictly rank local if, for every pair of graphs $G=(P,E)$ and $G'=(P,E')$, and for every pair of nodes x and y, if $B_G(x)=B_{G'}(x)$ and $B_G(y)=B_{G'}(y)$ then

$$A(G)[x] < A(G)[y] \Leftrightarrow A(G')[x] < A(G')[y]$$

  - the relative order of the nodes remains the same



- The InDegree algorithm is strictly rank local

Label Independence

- Label Independence: An algorithm is label independent if a permutation of the labels of the nodes yields the same permutation of the weights
  - the weights assigned by the algorithm do not depend on the labels of the nodes

Axiomatic characterization of the InDegree algorithm [BRRT05]

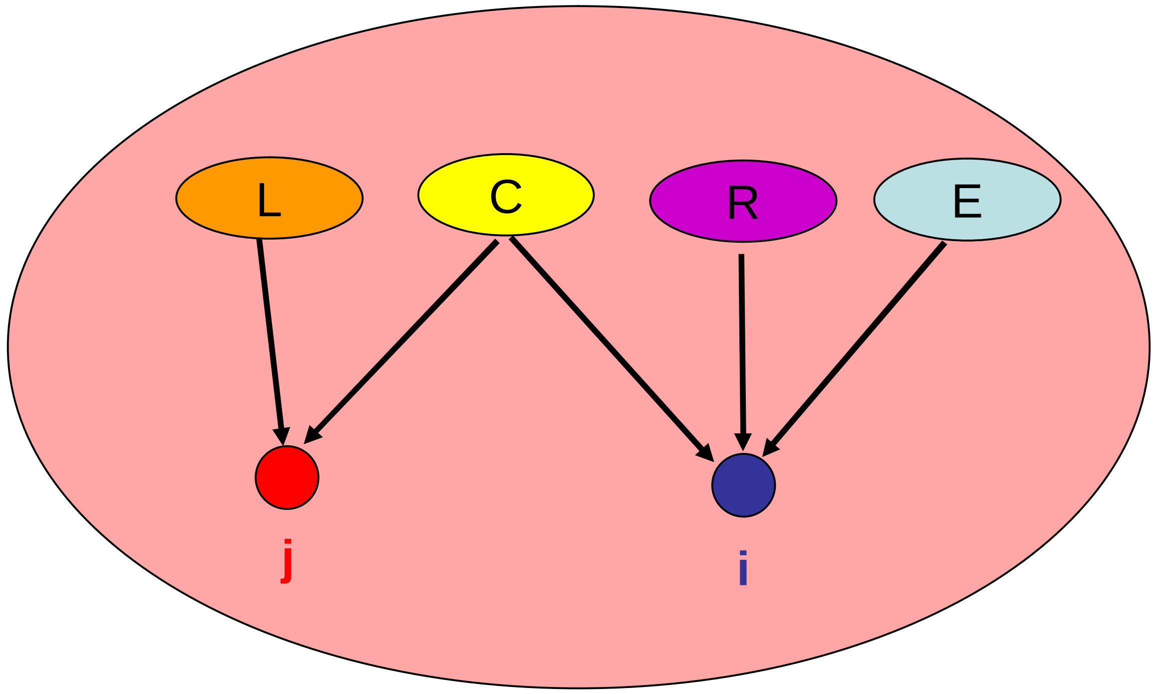
- Theorem: Any algorithm that is strictly rank local, strictly monotone and label independent is rank equivalent to the InDegree algorithm

Proof outline

- Consider two nodes i and j with $d(i) > d(j)$
- Assume that $w(i) < w(j)$

$|R| = |L|$

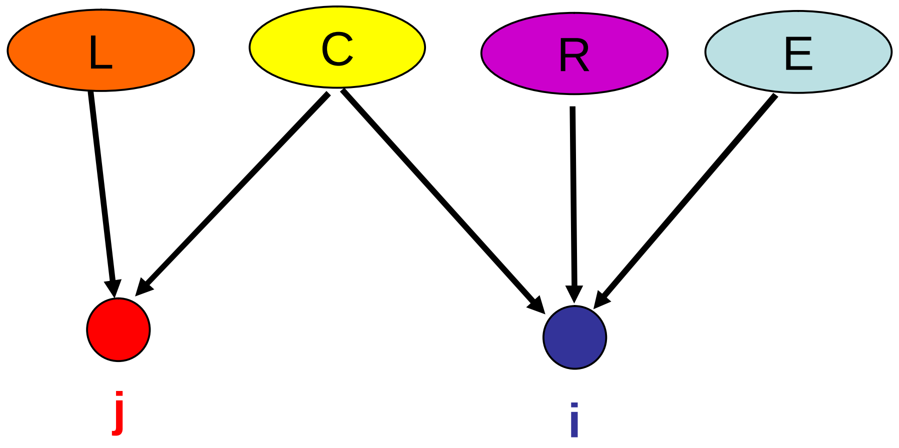
$|E| > 0$



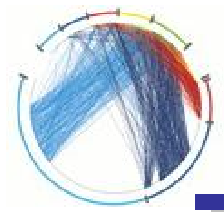
graph G

Proof outline

- Remove all links except to i and j
  - $w_1(i) < w_1(j)$ (from locality)

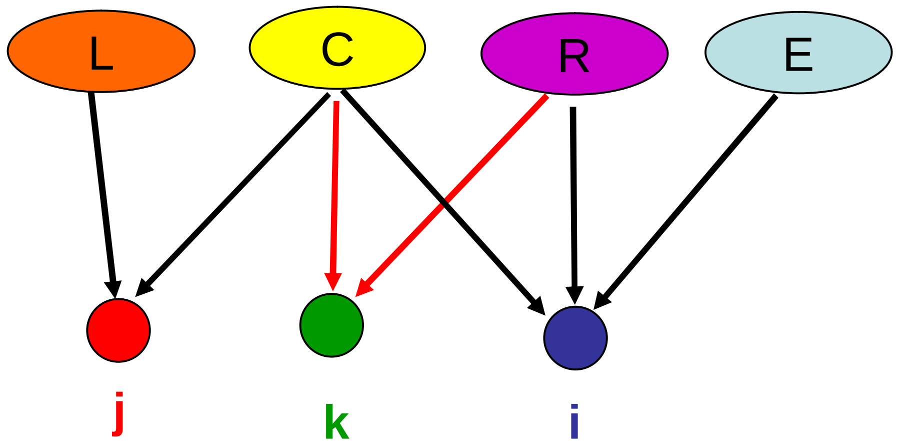


graph G_1



Proof outline

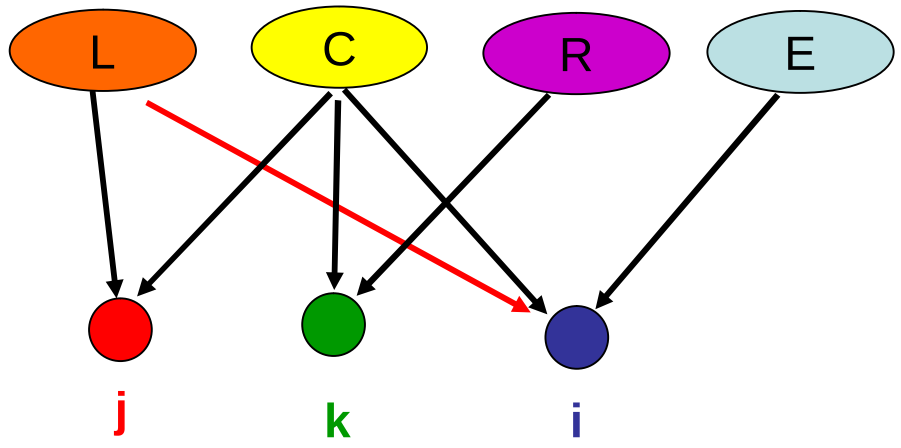
- Add links from C and R to node k
  - $w_2(i) < w_2(j)$ (from locality)
  - $w_2(k) < w_2(i)$ (from monotonicity)
  - $w_2(k) < w_2(j)$



graph G_2

Proof outline

- Remove links from R to i and add links from L to i
  - $w_3(k) < w_3(j)$ (from locality)

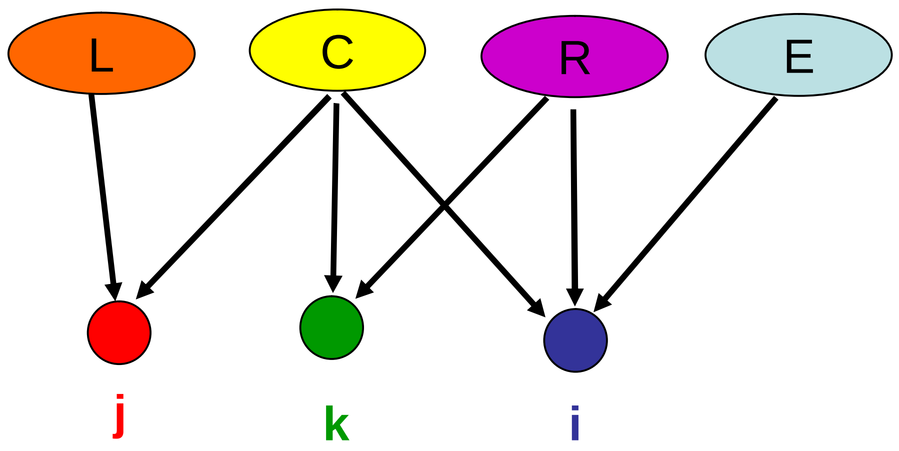


graph G_3

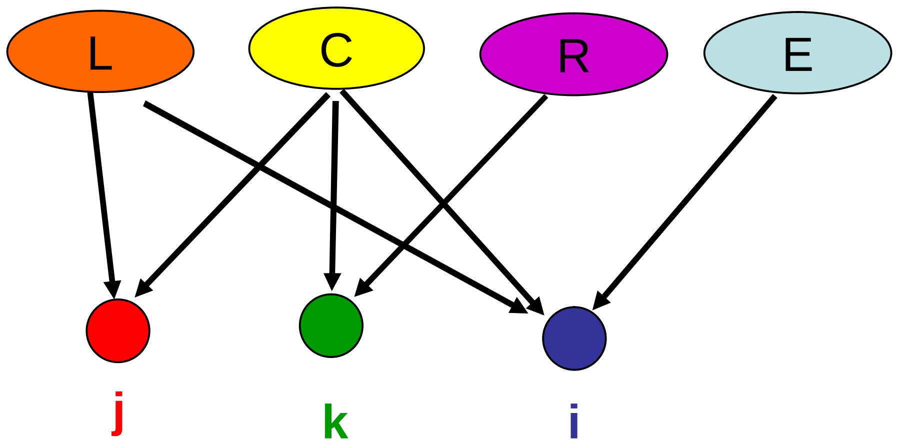
Proof outline

- Graphs G_2 and G_3 are the same up to a label permutation

$$L \leftrightarrow R$$
$$j \leftrightarrow k$$



graph G_2

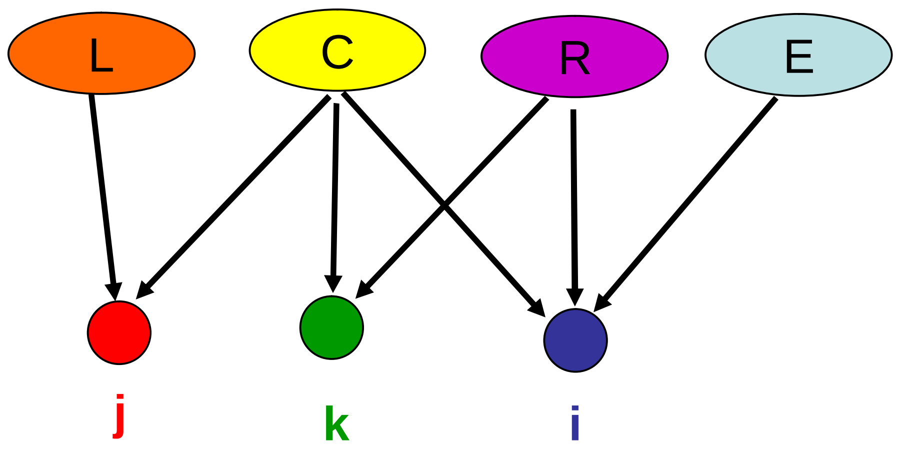


graph G_3

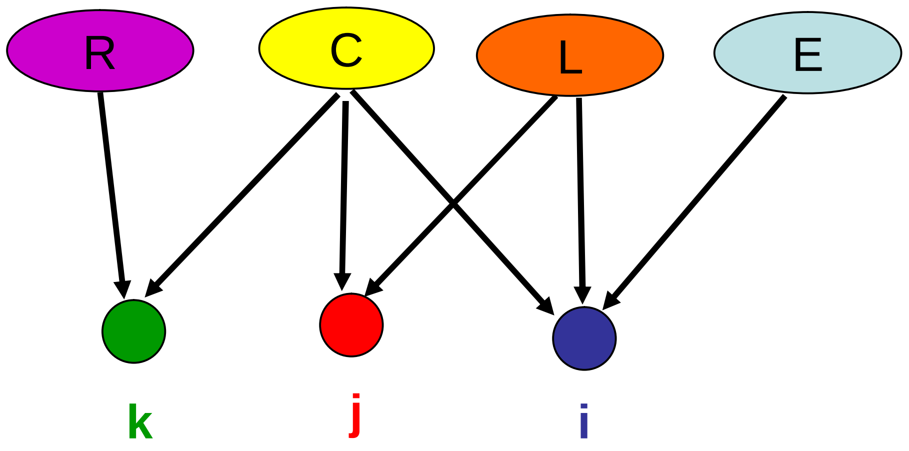
Proof outline

- Graphs G_2 and G_3 are the same up to a label permutation

$$L \leftrightarrow R$$
$$j \leftrightarrow k$$



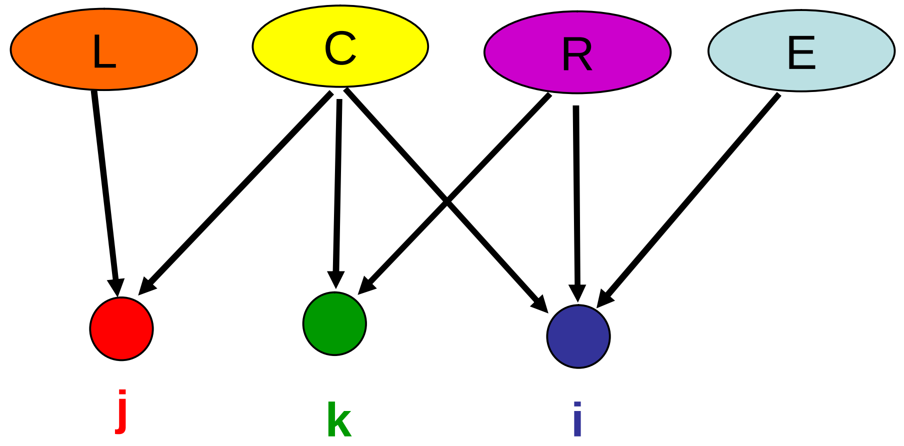
graph G_2



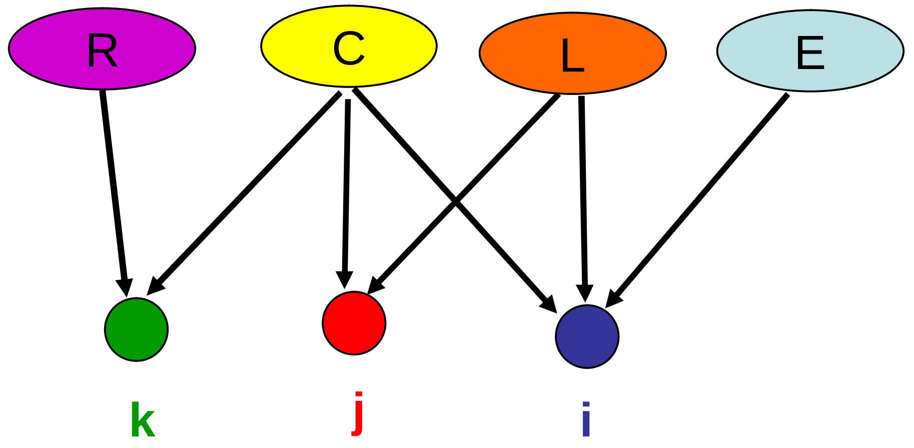
graph G_3

Proof outline

- We now have
  - $w_2(j) < w_2(k)$ and $w_3(j) < w_3(k)$ (shown before)
  - $w_2(j) = w_3(k)$ and $w_2(k) = w_3(j)$ (label independ.)
  - $w_2(j) > w_2(k)$ CONTRADICTION!



graph G_2



graph G_3

Axiomatic characterization

- All three properties are needed
  - locality
    - PageRank is also strictly monotone and label independent
  - monotonicity
    - consider an algorithm that assigns 1 to nodes with even degree, and 0 to nodes with odd degree
  - label independence
    - consider and algorithm that gives the more weight to links that come from some specific page (e.g. the Yahoo page)

Outline

- Axiomatic Characterizations of Link Analysis Ranking Algorithms
  - InDegree algorithm
  - PageRank algorithm
- Rank Aggregation
  - Computing aggregate scores
  - Computing aggregate rankings - voting

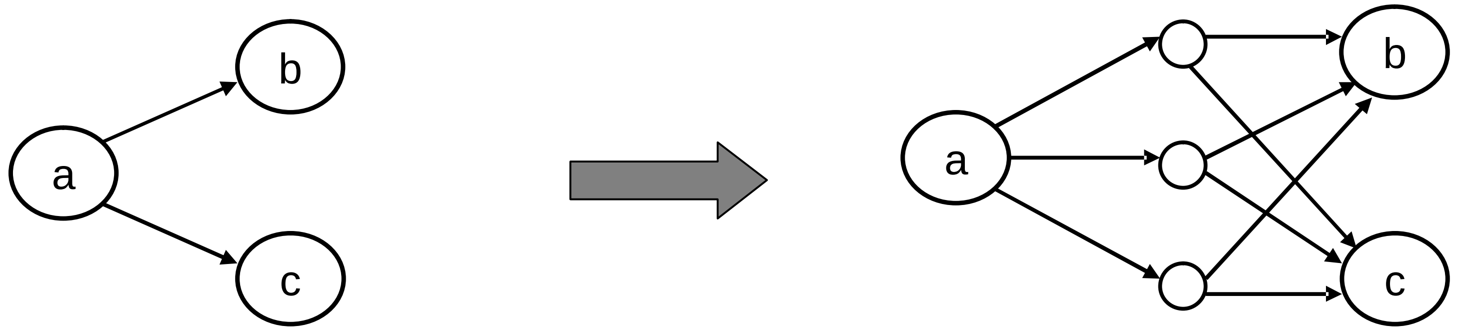
Self-edge axiom

§ Algorithm A satisfies the self-edge axiom if the following is true: If page a is ranked at least as high as page b in a graph $G(V,E)$, where a does not have a link to itself, then a should be ranked higher than b in $G(V,E \cup \{v,v\})$

Vote by committee axiom

- Algorithm A satisfies the vote by committee axiom if the following is true: If page *a* links to pages *b* and *c*, then the relative ranking of all the pages should be the same as in the case where the direct links from *a* to *b* and *c* are replaced by links from *a* to a new set of pages which link (only) to *b* and *c*

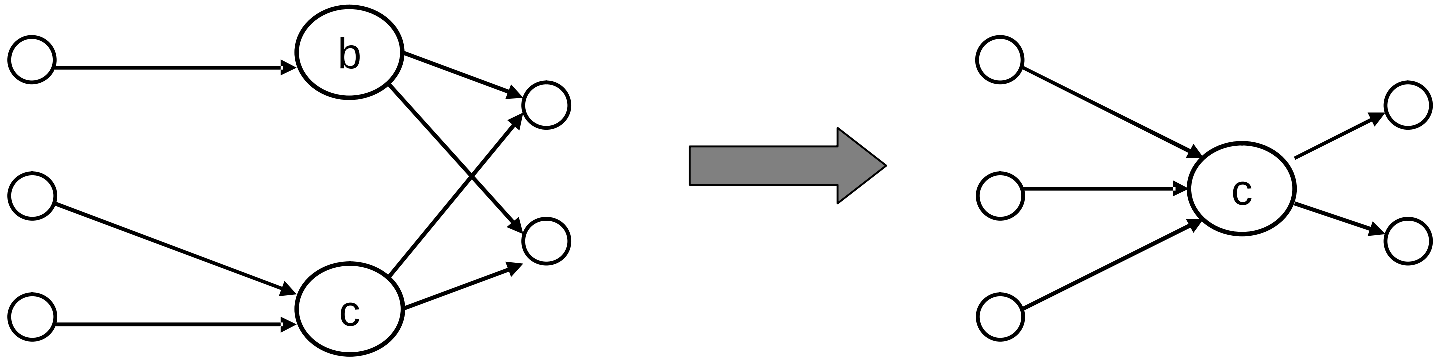
Vote by committee (example)



Collapsing axiom

- If there is a pair of pages a and b that link to the same set of pages, but the set of pages that link to a and b are disjoint, then if a and b are collapsed into a single page (a), where links of b become links of a, then the relative rankings of all pages (except a and b) should remain the same.

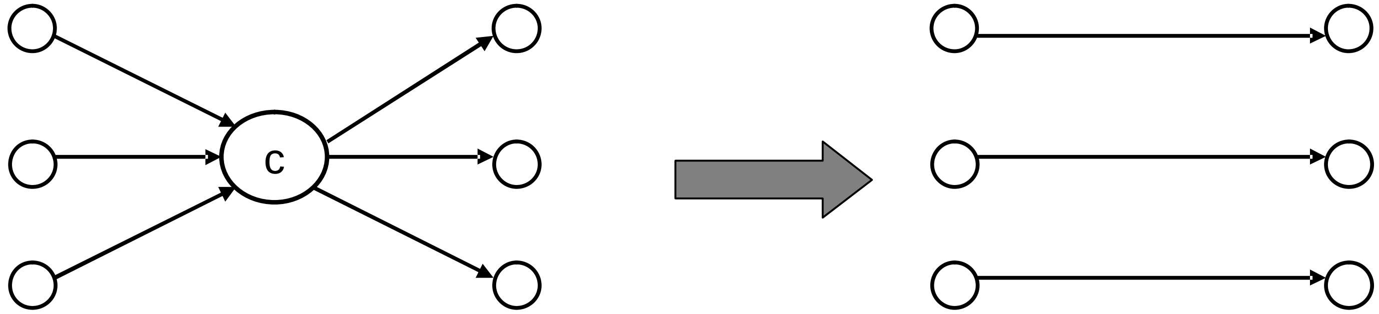
Collapsing axiom (example)



Proxy axiom

- If there is a set of k pages with the same importance that link to a, and a itself links to k other pages, then by dropping a and connect the pages in N(a) and P(a), the relative ranking of all pages (excluding a) should remain the same

Proxy axiom (example)



Axiomatic Characterization of PageRank Algorithm [AT04]

- The PageRank algorithm satisfies label independence, self-edge, vote by committee, collapsing and proxy axioms.

Outline

- Axiomatic Characterizations of Link Analysis Ranking Algorithms
  - InDegree algorithm
  - PageRank algorithm
- Rank Aggregation
  - Computing aggregate scores
  - Computing aggregate rankings - voting

Rank Aggregation

- Given a set of rankings $R_1, R_2, \ldots, R_m$ of a set of objects $X_1, X_2, \ldots, X_n$ produce a single ranking R that is in agreement with the existing rankings

Examples

- Voting
  - rankings $R_1, R_2, \ldots, R_m$ are the voters, the objects $X_1, X_2, \ldots, X_n$ are the candidates.

Examples

- Combining multiple scoring functions
  - rankings $R_1,R_2,\ldots,R_m$ are the scoring functions, the objects $X_1,X_2,\ldots,X_n$ are data items.
    - Combine the PageRank scores with term-weighting scores
    - Combine scores for multimedia items
      - color, shape, texture
    - Combine scores for database tuples
      - find the best hotel according to price and location

Examples

- Combining multiple sources
  - rankings $R_1,R_2,\ldots,R_m$ are the sources, the objects $X_1,X_2,\ldots,X_n$ are data items.
    - meta-search engines for the Web
    - distributed databases
    - P2P sources

Variants of the problem

- Combining scores
  - we know the scores assigned to objects by each ranking, and we want to compute a single score
- Combining ordinal rankings
  - the scores are not known, only the ordering is known
  - the scores are known but we do not know how, or do not want to combine them
    - e.g. price and star rating

Combining scores

- Each object X_i has m scores $(r_{i1},r_{i2},\ldots,r_{im})$
- The score of object X_i is computed using an aggregate scoring function $f(r_{i1},r_{i2},\ldots,r_{im})$

	R_1	R_2	R_3
X_1	1	0.3	0.2
X_2	0.8	0.8	0
X_3	0.5	0.7	0.6
X_4	0.3	0.2	0.8
X_5	0.1	0.1	0.1

Combining scores

- Each object X_i has m scores $(r_{i1}, r_{i2}, \ldots, r_{im})$
- The score of object X_i is computed using an aggregate scoring function $f(r_{i1}, r_{i2}, \ldots, r_{im})$
  - $f(r_{i1}, r_{i2}, \ldots, r_{im}) = \min\{r_{i1}, r_{i2}, \ldots, r_{im}\}$

	R_1	R_2	R_3	R
X_1	1	0.3	0.2	0.2
X_2	0.8	0.8	0	0
X_3	0.5	0.7	0.6	0.5
X_4	0.3	0.2	0.8	0.2
X_5	0.1	0.1	0.1	0.1

Combining scores

- Each object X_i has m scores $(r_{i1},r_{i2},\ldots,r_{im})$
- The score of object X_i is computed using an aggregate scoring function $f(r_{i1},r_{i2},\ldots,r_{im})$
  - $f(r_{i1},r_{i2},\ldots,r_{im}) = \max\{r_{i1},r_{i2},\ldots,r_{im}\}$

	R_1	R_2	R_3	R
X_1	1	0.3	0.2	1
X_2	0.8	0.8	0	0.8
X_3	0.5	0.7	0.6	0.7
X_4	0.3	0.2	0.8	0.8
X_5	0.1	0.1	0.1	0.1

Combining scores

- Each object X_i has m scores $(r_{i1},r_{i2},\ldots,r_{im})$
- The score of object X_i is computed using an aggregate scoring function $f(r_{i1},r_{i2},\ldots,r_{im})$
  - $f(r_{i1},r_{i2},\ldots,r_{im}) = r_{i1} + r_{i2} + \ldots + r_{im}$

	R_1	R_2	R_3	R
X_1	1	0.3	0.2	1.5
X_2	0.8	0.8	0	1.6
X_3	0.5	0.7	0.6	1.8
X_4	0.3	0.2	0.8	1.3
X_5	0.1	0.1	0.1	0.3

Top-k

- Given a set of n objects and m scoring lists sorted in decreasing order, find the top-k objects according to a scoring function f

- top-k: a set T of k objects such that $f(r_{j1},\ldots,r_{jm}) \leq f(r_{i1},\ldots,r_{im})$ for every object X_i in T and every object X_j not in T

- Assumption: The function f is monotone
  - $f(r_1,\ldots,r_m) \leq f(r_1',\ldots,r_m')$ if $r_i \leq r_i'$ for all i
- Objective: Compute top-k with the minimum cost

Cost function

- We want to minimize the number of accesses to the scoring lists
- Sorted accesses: sequentially access the objects in the order in which they appear in a list
  - cost C_s
- Random accesses: obtain the cost value for a specific object in a list
  - cost C_r

- If s sorted accesses and r random accesses minimize $s\,C_s + r\,C_r$

Example

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

- Compute top-2 for the sum aggregate function

Fagin’s Algorithm

1. Access sequentially all lists in parallel until there are k objects that have been seen in all lists

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin's Algorithm

1. Access sequentially all lists in parallel until there are k objects that have been seen in all lists

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin's Algorithm

1. Access sequentially all lists in parallel until there are k objects that have been seen in all lists

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin's Algorithm

1. Access sequentially all lists in parallel until there are k objects that have been seen in all lists

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin’s Algorithm

1. Access sequentially all lists in parallel until there are k objects that have been seen in all lists

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin’s Algorithm

2. Perform random accesses to obtain the scores of all seen objects

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin's Algorithm

3. Compute score for all objects and find the top-k

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

R	
X_3	1.8
X_2	1.6
X_1	1.5
X_4	1.3

Fagin's Algorithm

- X_5 cannot be in the top-2 because of the monotonicity property
  - $f(X_5) \leq f(X_1) \leq f(X_3)$

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

R	
X_3	1.8
X_2	1.6
X_1	1.5
X_4	1.3

Fagin’s Algorithm

- The algorithm is cost optimal under some probabilistic assumptions for a restricted class of aggregate functions

Threshold algorithm

1. Access the elements sequentially

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Threshold algorithm

1. At each sequential access
   a. Set the threshold t to be the aggregate of the scores seen in this access

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

t = 2.6

Threshold algorithm

1. At each sequential access
   b. Do random accesses and compute the score of the objects seen

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 2.6$

X_1	1.5
X_2	1.6
X_4	1.3

Threshold algorithm

1. At each sequential access
   c. Maintain a list of top-k objects seen so far

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 2.6$

X_2	1.6
X_1	1.5

Threshold algorithm

1. At each sequential access
   d. When the scores of the top-k are greater or equal to the threshold, stop

R_1	
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

R_2	
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

R_3	
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 2.1$

X_3	1.8
X_2	1.6

Threshold algorithm

1. At each sequential access
   d. When the scores of the top-k are greater or equal to the threshold, stop

R_1		R_2		R_3	
X_1	1	X_2	0.8	X_4	0.8
X_2	0.8	X_3	0.7	X_3	0.6
X_3	0.5	X_1	0.3	X_1	0.2
X_4	0.3	X_4	0.2	X_5	0.1
X_5	0.1	X_5	0.1	X_2	0

$t = 1.0$

X_3	1.8
X_2	1.6

Threshold algorithm

2. Return the top-k seen so far

R_1		R_2		R_3	
X_1	1	X_2	0.8	X_4	0.8
X_2	0.8	X_3	0.7	X_3	0.6
X_3	0.5	X_1	0.3	X_1	0.2
X_4	0.3	X_4	0.2	X_5	0.1
X_5	0.1	X_5	0.1	X_2	0

$t = 1.0$

X_3	1.8
X_2	1.6

Threshold algorithm

- From the monotonicity property for any object not seen, the score of the object is less than the threshold
  - $f(X_5) \leq t \leq f(X_2)$
- The algorithm is instance cost-optimal
  - within a constant factor of the best algorithm on any database

Combining rankings

- In many cases the scores are not known
  - e.g. meta-search engines – scores are proprietary information
- ... or we do not know how they were obtained
  - one search engine returns score 10, the other 100. What does this mean?
- ... or the scores are incompatible
  - apples and oranges: does it make sense to combine price with distance?

- In this cases we can only work with the rankings

The problem

- Input: a set of rankings $R_1,R_2,\ldots,R_m$ of the objects $X_1,X_2,\ldots,X_n$. Each ranking R_i is a total ordering of the objects
  - for every pair X_i,X_j either X_i is ranked above X_j or X_j is ranked above X_i

- Output: A total ordering R that aggregates rankings $R_1,R_2,\ldots,R_m$

Voting theory

- A voting system is a rank aggregation mechanism
- Long history and literature
  - criteria and axioms for good voting systems

What is a good voting system?

§ The Condorcet criterion

  § if object A defeats every other object in a pairwise majority vote, then A should be ranked first

§ Extended Condorcet criterion

  § if the objects in a set X defeat in pairwise comparisons the objects in the set Y then the objects in X should be ranked above those in Y

§ Not all voting systems satisfy the Condorcet criterion!

Pairwise majority comparisons

- Unfortunately the Condorcet winner does not always exist
  - irrational behavior of groups

	V_1	V_2	V_3
1	A	B	C
2	B	C	A
3	C	A	B

A > B B > C C > A

Pairwise majority comparisons

- Resolve cycles by imposing an agenda

	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D

Pairwise majority comparisons

- Resolve cycles by imposing an agenda

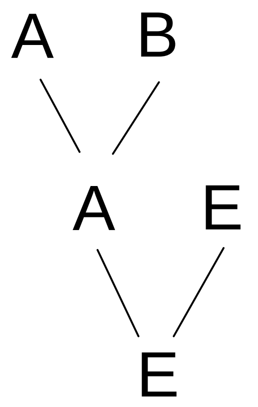
	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D

$$\begin{tikzcd}A \arrow[dr, dash] & & B \arrow[dl, dash] \\ & A & \end{tikzcd}$$

Pairwise majority comparisons

- Resolve cycles by imposing an agenda

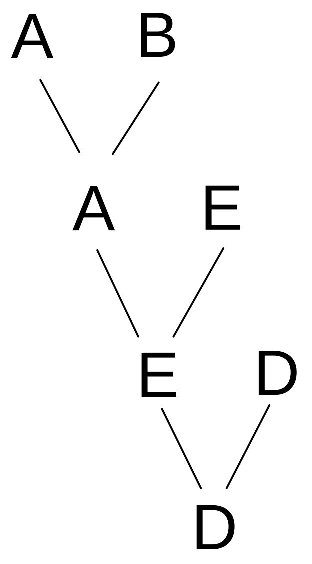
	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



Pairwise majority comparisons

- Resolve cycles by imposing an agenda

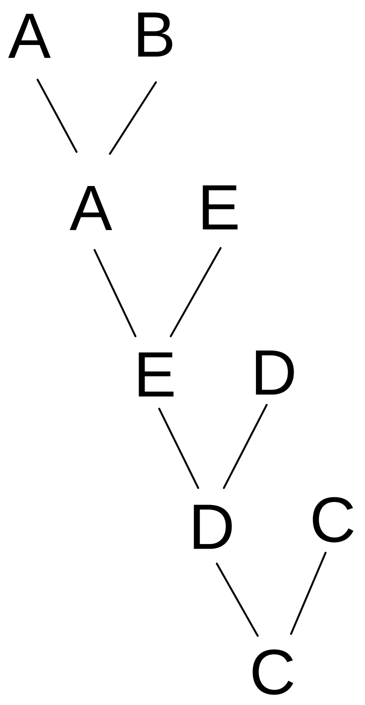
	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



Pairwise majority comparisons

- Resolve cycles by imposing an agenda

	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D

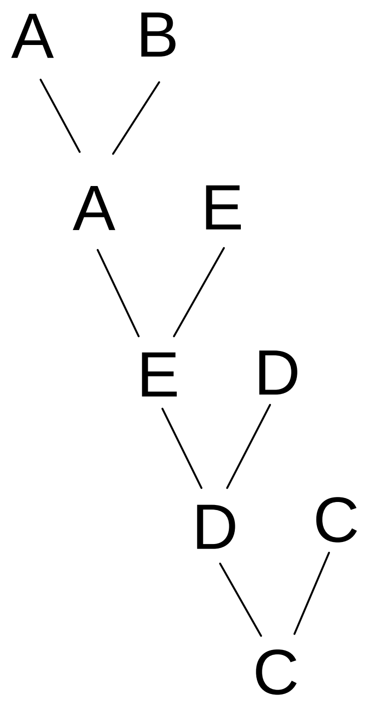


- C is the winner

Pairwise majority comparisons

- Resolve cycles by imposing an agenda

	V_1	V_2	V_3
1	A	D	E
2	B	E	A
3	C	A	B
4	D	B	C
5	E	C	D



- But everybody prefers A or B over C

Pairwise majority comparisons

- The voting system is not Pareto optimal
  - there exists another ordering that everybody prefers
- Also, it is sensitive to the order of voting

Plurality vote

- Elect first whoever has more 1st position votes

voters	10	8	7
1	A	C	B
2	B	A	C
3	C	B	A

- Does not find a Condorcet winner (C in this case)

Plurality with runoff

- If no-one gets more than 50% of the 1st position votes, take the majority winner of the first two

voters	10	8	7	2
1	A	C	B	B
2	B	A	C	A
3	C	B	A	C

first round: A 10, B 9, C 8
second round: A 18, B 9
winner: A

Plurality with runoff

- If no-one gets more than 50% of the 1st position votes, take the majority winner of the first two

voters	10	8	7	2
1	A	C	B	A
2	B	A	C	B
3	C	B	A	C

change the order of A and B in the last column

first round: A 12, B 7, C 8
second round: A 12, C 15
winner: C!

Positive Association axiom

- Plurality with runoff violates the positive association axiom

- Positive association axiom: positive changes in preferences for an object should not cause the ranking of the object to decrease

Borda Count

- For each ranking, assign to object X, number of points equal to the number of objects it defeats
    - first position gets n-1 points, second n-2, ..., last 0 points
- The total weight of X is the number of points it accumulates from all rankings

Borda Count

voters	3	2	2
1 (3p)	A	B	C
2 (2p)	B	C	D
3 (1p)	C	D	A
4 (0p)	D	A	B

A: 3*3 + 2*0 + 2*1 = 11p
B: 3*2 + 2*3 + 2*0 = 12p
C: 3*1 + 2*2 + 2*3 = 13p
D: 3*0 + 2*1 + 2*2 = 6p

BC
C
B
A
D

- Does not always produce Condorcet winner

Borda Count

- Assume that D is removed from the vote

voters	3	2	2
1 (2p)	A	B	C
2 (1p)	B	C	A
3 (0p)	C	A	B

A: 3*2 + 2*0 + 2*1 = 7p
B: 3*1 + 2*2 + 2*0 = 7p
C: 3*0 + 2*1 + 2*2 = 6p

BC
B
A
C

- Changing the position of D changes the order of the other elements!

Independence of Irrelevant Alternatives

- The relative ranking of X and Y should not depend on a third object Z
  - heavily debated axiom

Borda Count

- The Borda Count of an an object X is the aggregate number of pairwise comparisons that the object X wins
  - follows from the fact that in one ranking X wins all the pairwise comparisons with objects that are under X in the ranking

Voting Theory

- Is there a voting system that does not suffer from the previous shortcomings?

Arrow’s Impossibility Theorem

- There is no voting system that satisfies the following axioms
  - Universality
    - all inputs are possible
  - Completeness and Transitivity
    - for each input we produce an answer and it is meaningful
  - Positive Assosiation
  - Independence of Irrelevant Alternatives
  - Non-imposition
  - Non-dictatoriship

- KENNETH J. ARROW *Social Choice and Individual Values* (1951). Won Nobel Prize in 1972

Kemeny Optimal Aggregation

- Kemeny distance $K(R_1,R_2)$: The number of pairs of nodes that are ranked in a different order (Kendall-tau)
  - number of bubble-sort swaps required to transform one ranking into another
- Kemeny optimal aggregation minimizes

$$K(R,R_1,\ldots,R_m)=\sum_{i=1}^{m}K(R,R_i)$$

- Kemeny optimal aggregation satisfies the Condorcet criterion and the extended Condorcet criterion
  - maximum likelihood interpretation: produces the ranking that is most likely to have generated the observed rankings
- ...but it is NP-hard to compute
  - easy 2-approximation by obtaining the best of the input rankings, but it is not “interesting”

Locally Kemeny optimal aggregation

- A ranking R is locally Kemeny optimal if there is no bubble-sort swap that produces a ranking R' such that

  $K(R',R_1,\ldots,R_m) \leq K(R',R_1,\ldots,R_m)$

- Locally Kemeny optimal is not necessarily Kemeny optimal
- Definitions apply for the case of partial lists also

Locally Kemeny optimal aggregation

- Locally Kemeny optimal aggregation can be computed in polynomial time
  - At the i-th iteration insert the i-th element x in the bottom of the list, and bubble it up until there is an element y such that the majority places y over x

- Locally Kemeny optimal aggregation satisfies the Condorcet and extended Condorcet criterion

Rank Aggregation algorithm [DKNS01]

- Start with an aggregated ranking and make it into a locally Kemeny optimal aggregation
- How do we select the initial aggregation?
  - Use another aggregation method
  - Create a Markov Chain where you move from an object X, to another object Y that is ranked higher by the majority

Spearman's footrule distance

- Spearman's footrule distance: The difference between the ranks R(i) and R'(i) assigned to object i

$$F(R,R') = \sum_{i=1}^{n} |R(i) - R'(i)|$$

- Relation between Spearman's footrule and Kemeny distance

$$K(R,R') \leq F(R,R') \leq 2K(R,R')$$

Spearman’s footrule aggregation

- Find the ranking R, that minimizes

$$F(R, R_1, h, R_m) = \sum_{i=1}^{m} F(R, R_i)$$

- The optimal Spearman’s footrule aggregation can be computed in polynomial time
  - It also gives a 2-approximation to the Kemeny optimal aggregation

- If the median ranks of the objects are unique then this ordering is optimal

Example

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	A
3	C
4	D

A: $(1 , \boxed{2} , 3)$
B: $(1 , \boxed{1} , 2)$
C: $(3 , \boxed{3} , 4)$
D: $(3 , \boxed{4} , 4)$

The MedRank algorithm

- Access the rankings sequentially

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	
2	
3	
4	

The MedRank algorithm

- Access the rankings sequentially
  - when an element has appeared in more than half of the rankings, output it in the aggregated ranking

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	
3	
4	

The MedRank algorithm

- Access the rankings sequentially
  - when an element has appeared in more than half of the rankings, output it in the aggregated ranking

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	A
3	
4	

The MedRank algorithm

- Access the rankings sequentially
  - when an element has appeared in more than half of the rankings, output it in the aggregated ranking

R_1	
1	A
2	B
3	C
4	D

R_2	
1	B
2	A
3	D
4	C

R_3	
1	B
2	C
3	A
4	D

R	
1	B
2	A
3	C
4	

The MedRank algorithm

- Access the rankings sequentially
  - when an element has appeared in more than half of the rankings, output it in the aggregated ranking

R_1		R_2		R_3		R	
1	A	1	B	1	B	1	B
2	B	2	A	2	C	2	A
3	C	3	D	3	A	3	C
4	D	4	C	4	D	4	D

The Spearman’s rank correlation

- Spearman’s rank correlation

$$S(R,R') = \sum_{i=1}^{n} \left(R(i) - R'(i)\right)^2$$

- Computing the optimal rank aggregation with respect to Spearman’s rank correlation is the same as computing Borda Count
  - Computable in polynomial time

Extensions and Applications

- Rank distance measures between partial orderings and top-k lists
- Similarity search
- Ranked Join Indices
- Analysis of Link Analysis Ranking algorithms
- Connections with machine learning

References

- A. Borodin, G. Roberts, J. Rosenthal, P. Tsaparas, Link Analysis Ranking: Algorithms, Theory and Experiments, ACM Transactions on Internet Technologies (TOIT), 5(1), 2005
- Ron Fagin, Ravi Kumar, Mohammad Mahdian, D. Sivakumar, Erik Vee, Comparing and aggregating rankings with ties , PODS 2004
- M. Tennenholtz, and Alon Altman, "On the Axiomatic Foundations of Ranking Systems", Proceedings of IJCAI, 2005
- Ron Fagin, Amnon Lotem, Moni Naor. Optimal aggregation algorithms for middleware, J. Computer and System Sciences 66 (2003), pp. 614-656. Extended abstract appeared in Proc. 2001 ACM Symposium on Principles of Database Systems (PODS '01), pp. 102-113.
- Alex Tabbarok Lecture Notes
- Ron Fagin, Ravi Kumar, D. Sivakumar Efficient similarity search and classification via rank aggregation, Proc. 2003 ACM SIGMOD Conference (SIGMOD '03), pp. 301-312.
- Cynthia Dwork, Ravi Kumar, Moni Naor, D. Sivakumar. Rank Aggregation Methods for the Web. 10th International World Wide Web Conference, May 2001.
- C. Dwork, R. Kumar, M. Naor, D. Sivakumar, "Rank Aggregation Revisited," WWW10; selected as Web Search Area highlight, 2001.