Coordination Mechanisms for Scheduling

Yossi Azar

Tel-Aviv University

ADFOCS 2009 (lectures 1+2)

Price on Anarchy
[Koutsoupias-Papadimitiriou]

- Selfish users 
- User goal: minimize its cost
- Nash Equilibrium (NE) 
- System goal (e.g. Social welfare)
- The worst ratio of *NE* cost to *OPT* cost

Price of Anarchy Concept

- Not algorithmic
- Only analysis
- What to do if PoA is large
- How to influence the system

Possible Solutions

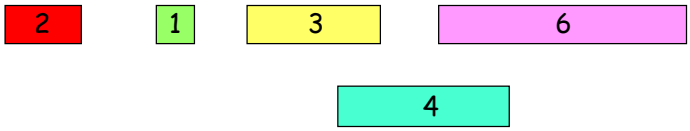
- Change the system (add tolls, payments)
- Stackelberg strategy = control some users
- *Disadvantages:* changing the settings, global knowledge
- *Challenge:* influence within the same setting and locally (distributed)

Coordination Mechanism
[Christodoulou-Koutsoupias-Nanavati]

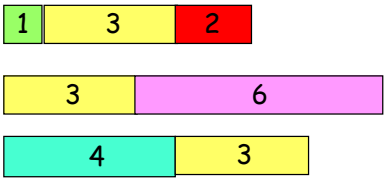
- *Mechanism*: local policy (algorithm) that assigns a cost for each strategy of the user
- *Advantages*: local, same type of cost
- *Goal*: achieving good NE
- *Example*: scheduling jobs on machines

Coordination Mechanism for Scheduling

- *Jobs* are agents

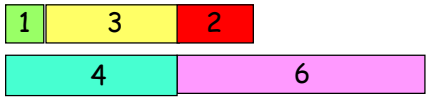


- Each job chooses a machine
- *Policy* for each machine which decides how to schedule jobs assigned to it



Coordination Mechanism for Scheduling

- A policy induces cost on each job if assigned to machine
- Each Policy induces *NE* on jobs



- *Local policy* – depends on jobs assigned to machine

Pure-strategy vs. Mixed-strategy NE

- Existence is proved for mixed NE
- Pure NE is more attractive
- With pure NE we can talk about convergence
- It is easier to justify the measure

- We focus of Pure NE

Unrelated Machine Scheduling

- m unrelated machines 
- n jobs – each **owned** by different user 
- p_{ij} - processing time of job i on machine j
- *System goal*: minimize completion time
- *User goal*: minimize its own completion time

Identical Machine Scheduling

- m identical machines
- n jobs – each **owned** by different user
- p_i - processing time of job i
- *System goal*: minimize completion time
- *User goal*: minimize its own completion time

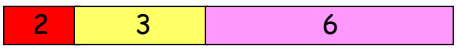
Challenge

Design policies that results in

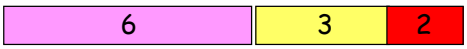
good NE (i.e. low PoA)

Local Scheduling Policies (cont)

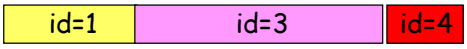
- Shortest First Policy



- Longest First Policy

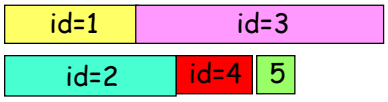


- I.D. priorities Policy 1 > 2 > 3 … > n



I.D. Priority

- *Observation*: NE for I.D. priority for all machines → Greedy online algorithm
- Greedy = Assign job to machine that the job would complete first



I.D. Priority (cont)

- *Proof*: The first job does not care about any other job. Continue by induction

Conclusions:

- Pure NE always exists
- Convergence always exists
- Easy computation of NE
- *Remark* – not true if I.D. priority ordering is different for each machine

I.D. Priority (cont)

- *Conclusion:* PoA of I.D priority
  = Competitive ratio of greedy
- *Claim:* Greedy is 2-1/m competitive for identical machines
- *Proof:* …

Shortest First Policy

- *Observation:* Shortest First Policy for all machines → Greedy Increasing Sizes
- Pure NE + convergence always exists
- *Conclusion:* PoA of Shortest First
  = Approx ratio of greedy increasing
- *Claim:* Greedy Increasing is 2-1/m approx for identical machines
- *Proof:* …

Makespan Policy

- Delay all jobs until machine completes

(not an ordering policy)

Observation: Makespan Policy for all machines → Greedy Local Search

Observation: Pure NE + convergence always exists

Makespan Policy

Proof: Consider the load vector sorted in non-increasing order. It always improves lexicographically

- *Claim:* Local search is 2-2/(m+1) approx for identical machines
- *Proof:* …

Longest First Policy

- *Observation:* Longest First Policy for all machines → Greedy Decreasing Sizes
- Pure NE + convergence always exists
- *Conclusion:* PoA of of Longest First
  = Approx ratio of greedy decreasing

- *Claim :* Greedy Decreasing is 4/3 approx for identical machines
- *Proof:* …

Related Machine Scheduling

- ***m*** machines, machine j has speed v_j
- ***n*** jobs – each **owned** by different user

- processing time of job i on machine j

$$p_i / v_j$$

- *System goal*: minimize completion time
- *User goal*: minimize its own completion time

I.D. Priority

- *Recall:*

  PoA of I.D. priority

  = Competitive ratio of greedy

- *Claim :* Greedy is **$\Theta(\log m)$** competitive for ***related*** machines

- Same for *Shortest first* (greedy increasing)

Makespan Policy

- *Recall:*

  PoA of I.D. priority

  = Approx ratio of Greedy Local Search

- *Claim :* Local Search is **$\Theta(\log m / \log\log m)$** approx for ***related*** machines

- *Proof:* … (lower bound)

Longest First Policy

- *Recall:*

  PoA of Longest First

  = Approx ratio of greedy decreasing

- *Claim :* Greedy (decreasing) is **1.52-1.59** approx for ***related*** machines

- *Proof:* …

Restricted Assignment (bipartite) Machine Scheduling

- ***m*** machines
- ***n*** jobs – job i has set S_i allowed machines

- processing time of job i on machine j

  p_i **if** $j \in S_i$ otherwise ∞

- *System goal*: minimize completion time
- *User goal*: minimize its own completion time

I.D. or Shortest or Longest First

- *Recall:*

PoA of I.D. priority

= Competitive ratio of greedy

- *Claim :* Greedy is **Θ(log m)** competitive for ***restricted assignment*** model
- *Proof:* …

MakeSpan Policy

- Recall: Delay all jobs until machine completes

(→ corresponds to local search)

- *Claim:* Local search is **Θ**(log m / loglog m) approx for ***restricted assignment***

- *Proof:* …(special case + exercise)

Unrelated Machine Scheduling

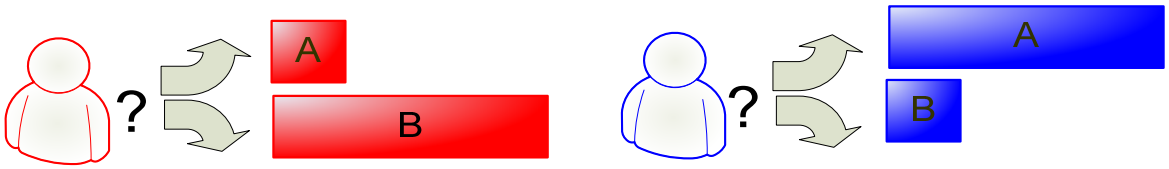
- *m* unrelated machines

- *n* jobs – each **owned** by different user

- p_{ij} - processing time of job i on machine j

- *System goal*: minimize completion time
- *User goal*: minimize its own completion time

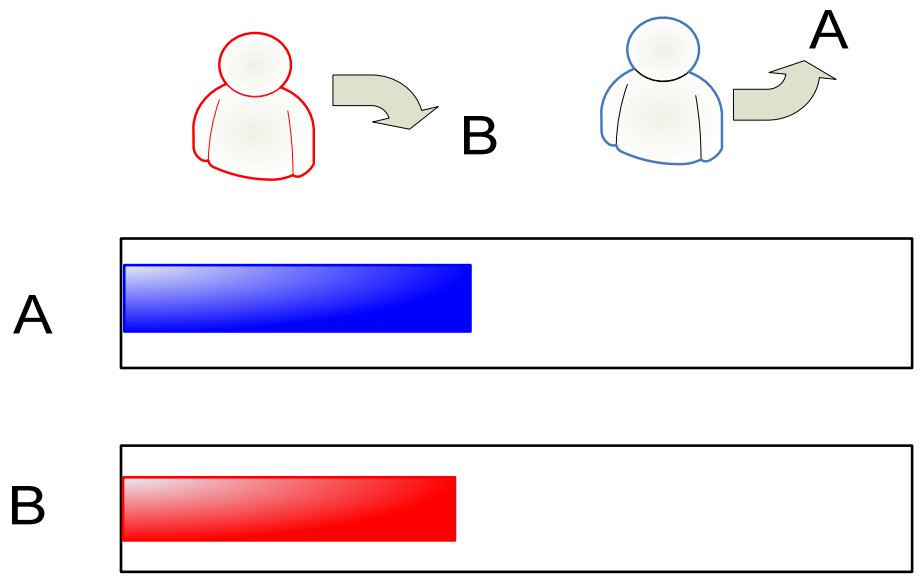
Unrelated Machines Scheduling



Machine A

Machine B

Equilibrium for Longest First



PoA of Longest First

- Results in poor NE
- The PoA is unbounded even for 2 machines
- Same for Makespan policy

PoA of I.D. or Shortest First

- PoA of both is at most m
- *Proof:* (exercise)

- PoA of I.D is at least m
- *Proof:* …

- PoA of Shortest First is at least m

Question ?

- Is there a policy with "reasonable" PoA

Type of Policies

- *Local policy* – depends on jobs assigned to machine
- *Strongly local policy* - depends only on processing time of jobs on that machine

- *Ordering Policy* = IIA (independence of irrelevant alternative)

Negative Results (strongly local)

- PoA of any strongly local ordering policy

  is at least **m/2**

Positive Results (local):

- Local policy with PoA of **O(log m)**

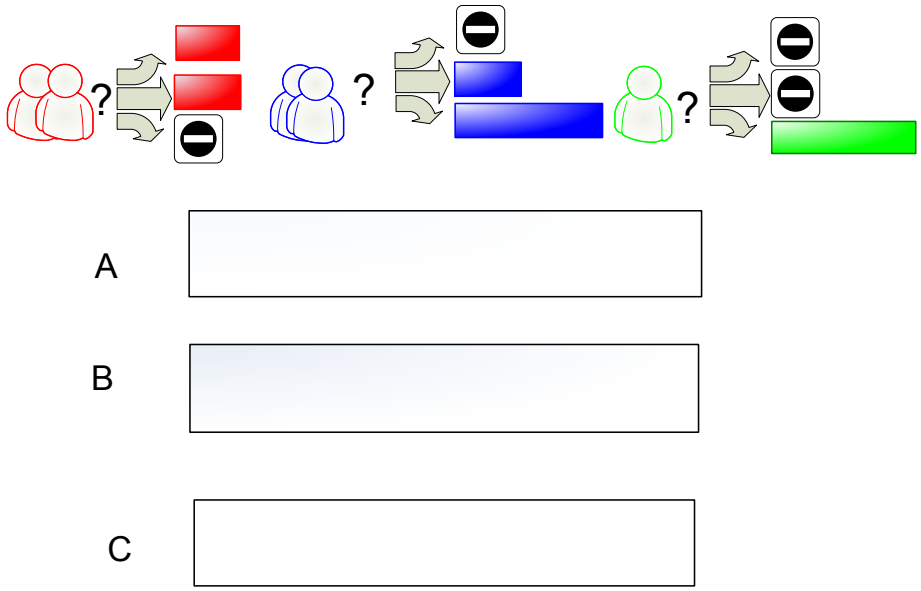
Lower Bound for Strongly Local Policy

- We construct an instance that all jobs on the same machine has (almost) the same processing time
- It is hard for a machine to distinguish between jobs
- There is a way to break ties such that PoA is **m/2**
- Given an arbitrary ordering function for each machine we carefully choose I.D. or sizes of jobs to behave as in the instance above

Idea of the Proof

- m types of jobs
- Type j can be scheduled on machines j & j+1
- Processing time of type j on machine j is low and on machine j+1 is high (ratio is j)
- All jobs on machine j have almost the same processing time

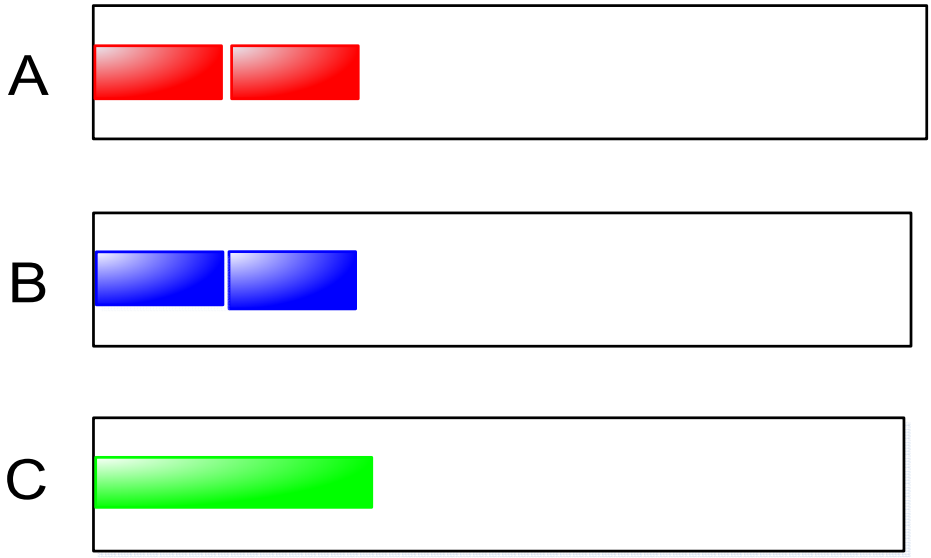
Example for Shortest-First



Idea of the Proof

- OPT assign all jobs of type j to machine j
- Number of jobs is chosen such that OPT has the same completion time for all machines

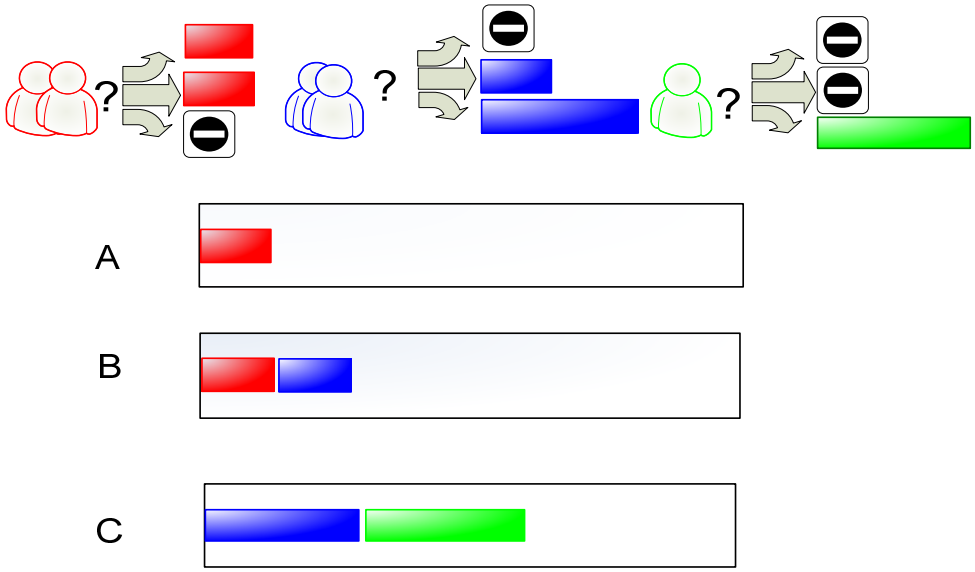
Optimal Assignment



Idea of the Proof

- In NE about half jobs of type j are on machine j and half on machine j+1
- Completion time of NE grows linearly in m

Equilibrium for Shortest-First



Coordination Mechanism for Unrelated Machine Scheduling

- *m* unrelated machines

- *n* jobs – each **owned** by different user

- p_{ij} - processing time of job i on machine j

- *System goal*: minimize completion time
- *User goal*: minimize its own completion time

Efficiency Based Algorithm

- Order jobs on each machine by their efficiency
- Efficiency of job on machine is:

  The ratio between job's best processing time to its processing time on this machine
- PoA of algorithm is O(log m)

Idea of the Proof

- Volume of jobs is not fixed
- Min-Volume: smallest possible volume
- Min-Volume assignment can be far from OPT
- The proof works through Min-Volume of the Efficiency based assignment

Idea of the Proof

- Remaining Volume above height h = min-volume of jobs assigned above height h
- Key Lemma: Remaining volume halved when the height increases by OPT

Efficiency Based Algorithm

- Unfortunately – pure NE may not exist
- Iterative improvement may cycle
- Modified algorithm guarantees convergence and pure NE with PoA of $O(\log^2 m)$

Modified Algorithm

- Each machine simulate log m submachines (by round robin)
- Submachine k of machine j handles jobs on efficiency between 2^{-k} and 2^{-k+1}
- Jobs are ordered on submachine by Shortest-First
- PoA of algorithm is $O(\log^2 m)$

Achieving PoA of log m

- Competitive analysis comes to help
- It is not a simple Greedy online algorithm

Competitive Algorithm for Unrelated Machine Scheduling

- *m* unrelated machines

- *n* jobs – jobs arrive one by one
- Jobs must be assigned immediately

- p_{ij} - processing time of job i on machine j
- *System goal*: minimize completion time
- *Measure*: Competitive ratio

More General Goal Function

- Minimize the L_p norm (we need $p=\infty$)
- Actually $p=O(\log m)$ is good approximation for $p=\infty$

  Proof:

- Greedy algorithm : assign job to minimize the L_p norm or the resulting vector

  Exact Formula:...

Competitive Online Scheduling

- *Claim:* Greedy algorithm is O(p) competitive
- *Proof:*

How to make it coordination Mechanism

- Give I.D priority on jobs
- Cost of job on machine is defined by looking only on jobs on that machine with higher priority
- Cost of job – cost of online algorithm SCALED appropriately

Search Ranking Problems

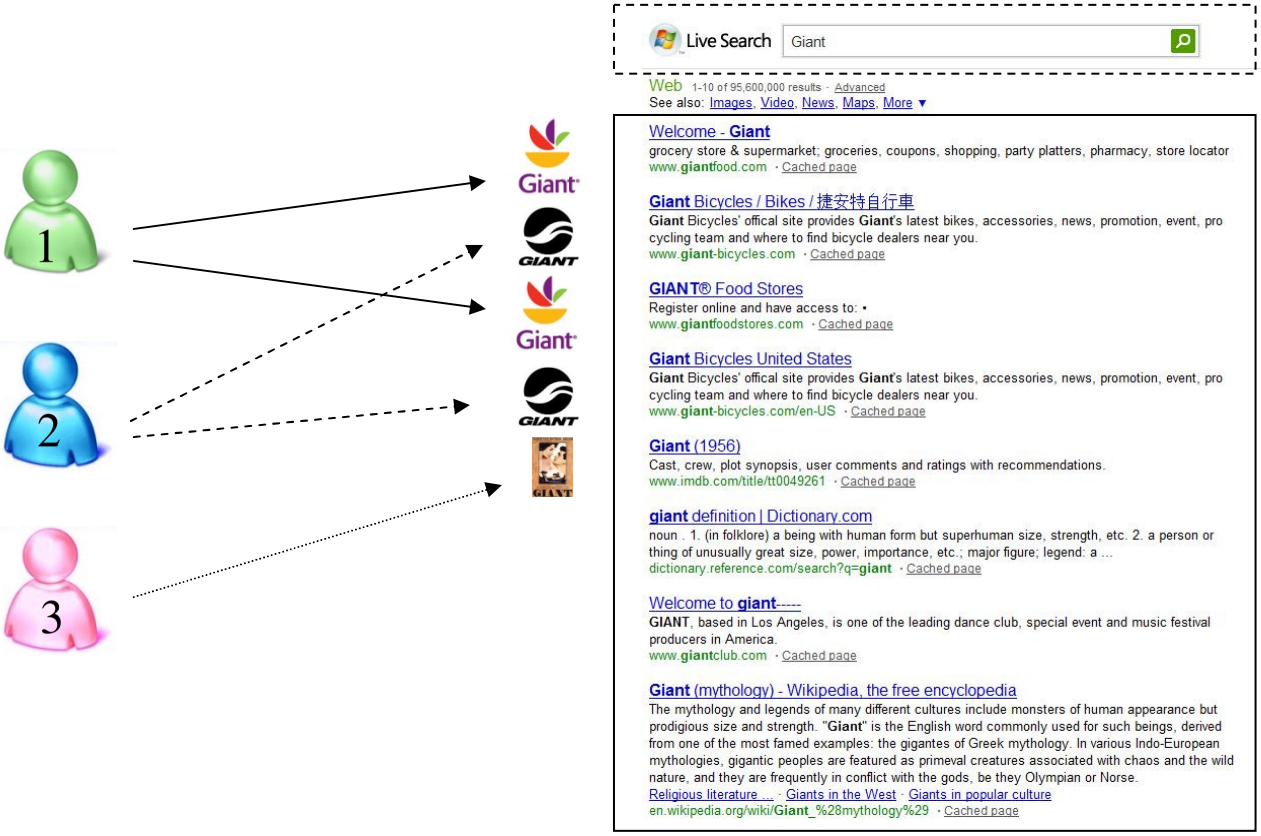
Yossi Azar

Tel Aviv University

ADFOCS 2009 (lecture 3)

Web search and ranking

How to find and rank relevant web pages for a query?



Web search and ranking

- Basic ranking approaches based on:
  - page contents
  - hyperlink structure of the web

- The relevance of results to specific user is not clear.

- Recent re-ranking approaches based on user logs.

Re-ranking based on user logs

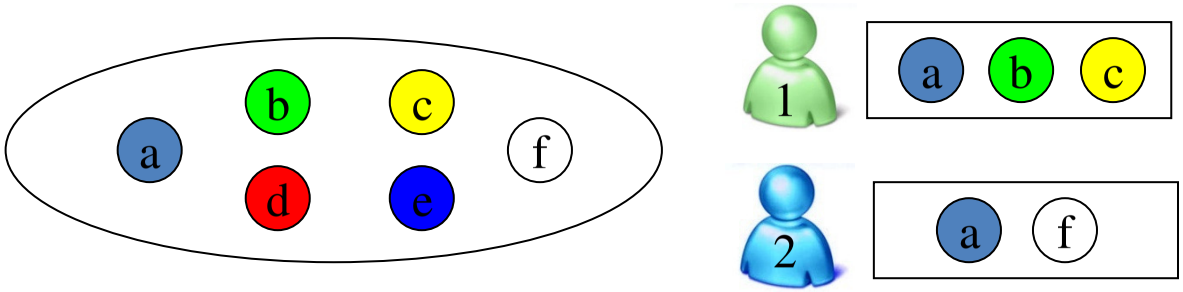
- Classic studies assume users have same preference over results. This is clearly not true…

  – A  that looks for a bike clearly prefers to see GIANT BICYCLES before any Giant.

A theoretical re-ranking model

- Input:
  - collection of result pages.
  - users types:
    - subset of pages.
    - (search behavior)

- Goal: rank pages to minimize average effort of users.



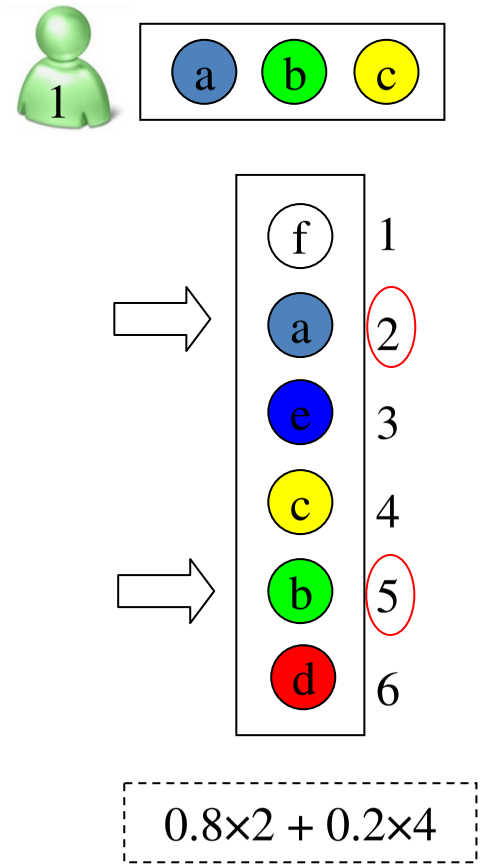
effort ?!

User effort

- What is the effort of a user type?
  - looking for first relevant result?
  - looking for all relevant results?
  - maybe user type is more complex?

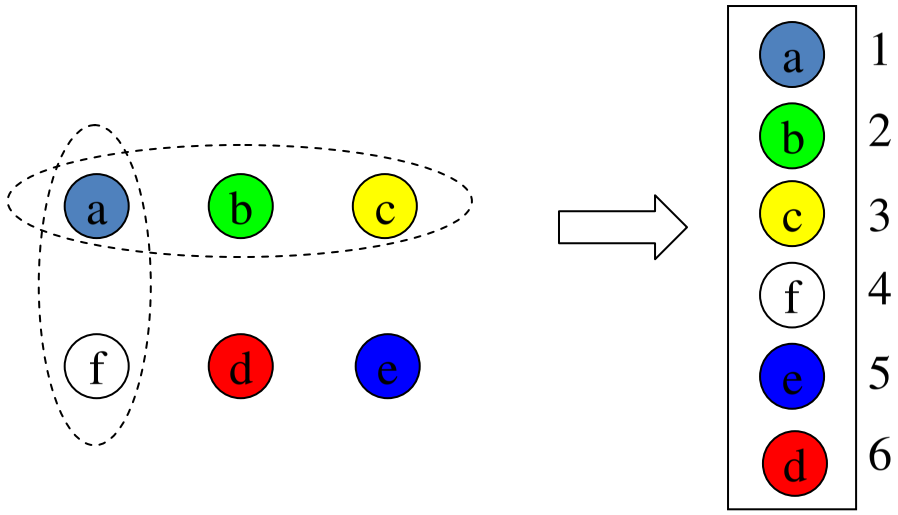
    e.g., must see first relevant result but with 20% also the second one?

- In practice, all options exists…
  - navigational - first result is relevant.
  - informational - all results are relevant.

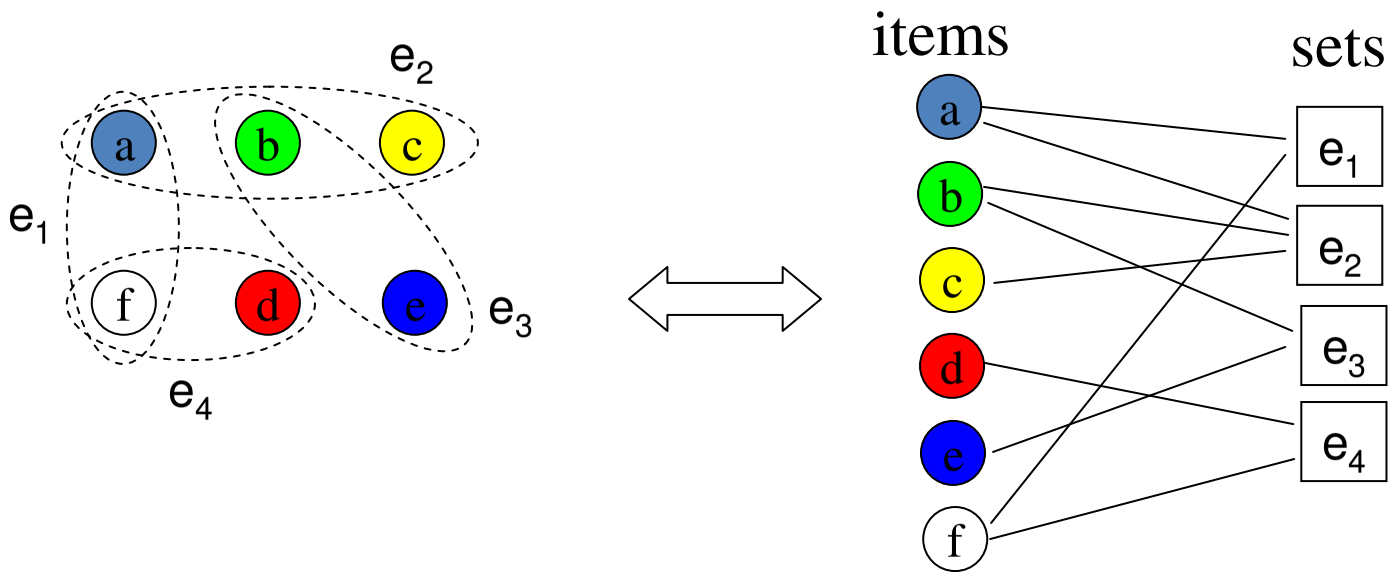


Graph-theoretic problem definition

- Input:
  - vertices = result pages.
  - hyperedges + weights = users.
- Goal: find an ordering of vertices that minimizes the weighted cover time of the hyperedges.



Cover time: First vertex in Hyperedge



– this is the min-sum variant of set cover: order items to minimize sum of covering times of sets.

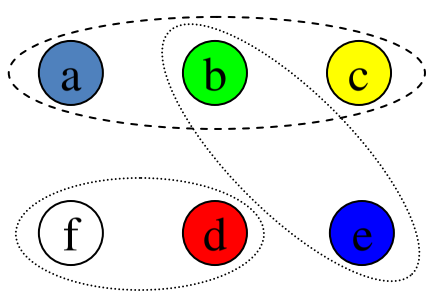
Set Cover vs. Min-Sum set cover

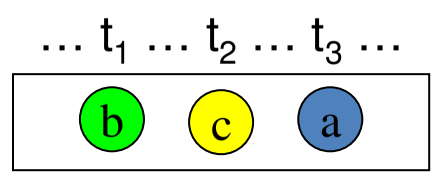
- Greedy algorithm: Iteratively choose an element that covers largest number of uncovered sets
- Greedy is $\log n$ approximation for set cover
- Greedy is 4 approximation for min-sum set cover
- NP-hard to approximate within 4-ε.
- [Feige, Lovasz, Tetali '04]

Greedy approximation analysis

- Some notation:
  - w(e) - the weight of hyperedge *e*.
  - $t_{OPT}(e)$ - the time that OPT covered *e*.
  - $t_{ALG}(e)$ - the time that ALG covered *e*

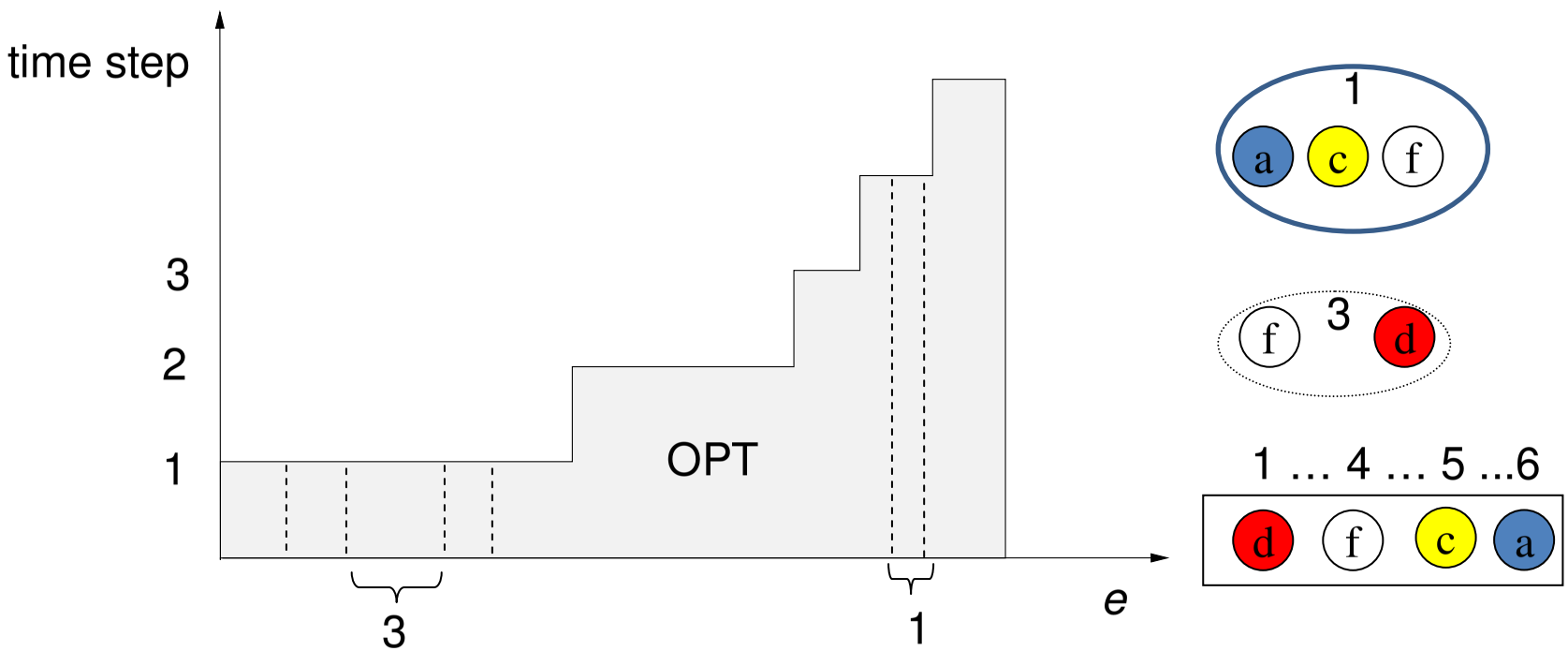
$$OPT = \sum_e w(e) \times t_{OPT}(e)$$





Histogram for the optimal solution

– create a column of height = $t_{OPT}(e)$ and width = $w(e)$, for every e, and order them by time.



Histogram for the greedy solution

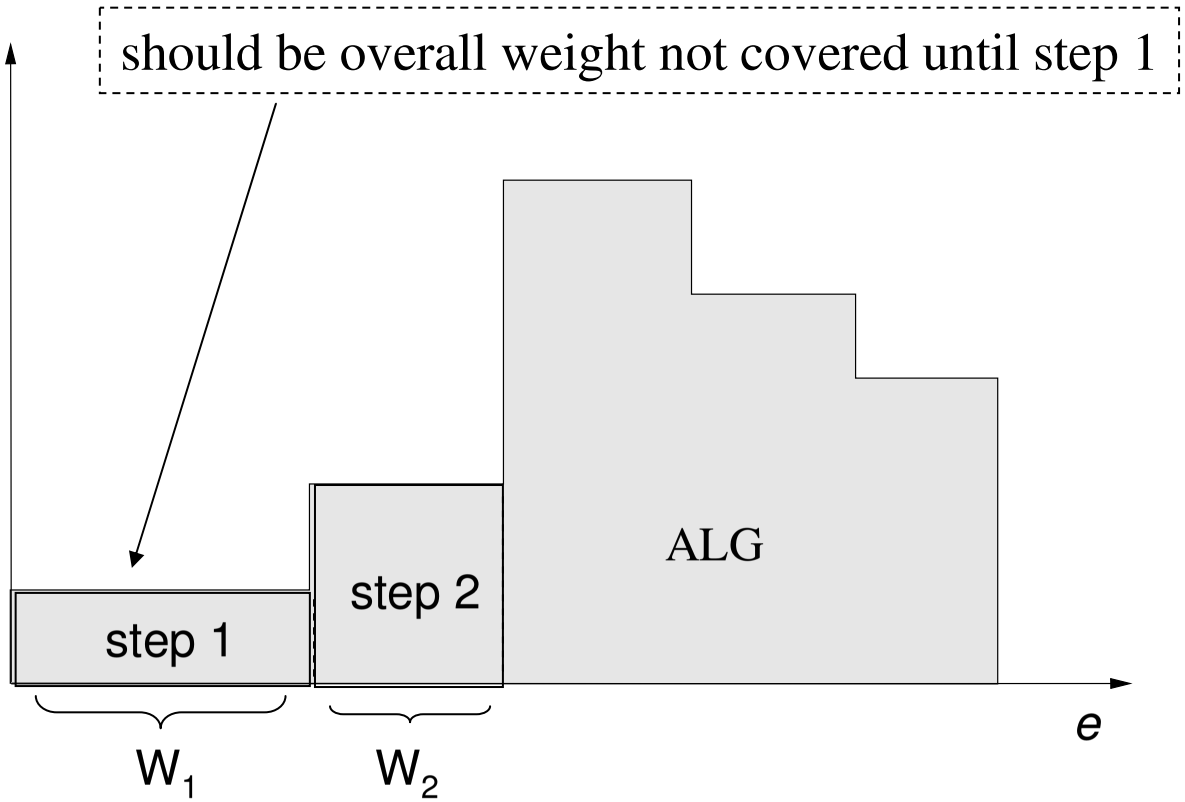
- recall that $\mathrm{ALG} = \sum_{e} w(e) \times t_{ALG}(e)$

$$= \sum_{t} \sum_{e: t \le t_{ALG}(e)} w(e)$$

overall remaining weight
(not covered until step t).

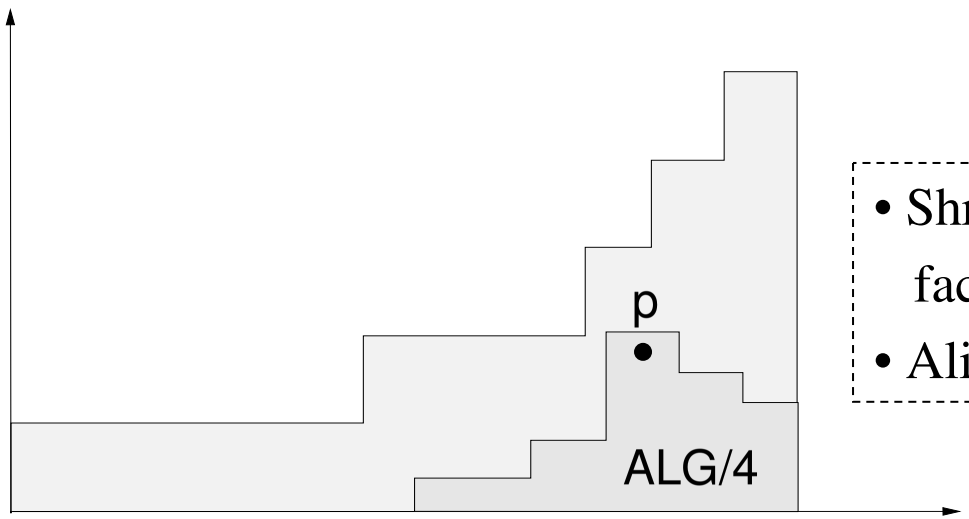
- create a column of width = w(e), and order by time…
- the area of all columns that are part of step t is exactly overall remaining weight of step t.

Histogram for the greedy solution



$$\text{height}_t = \frac{\text{overall remaining weight}}{\text{weight covered}}$$

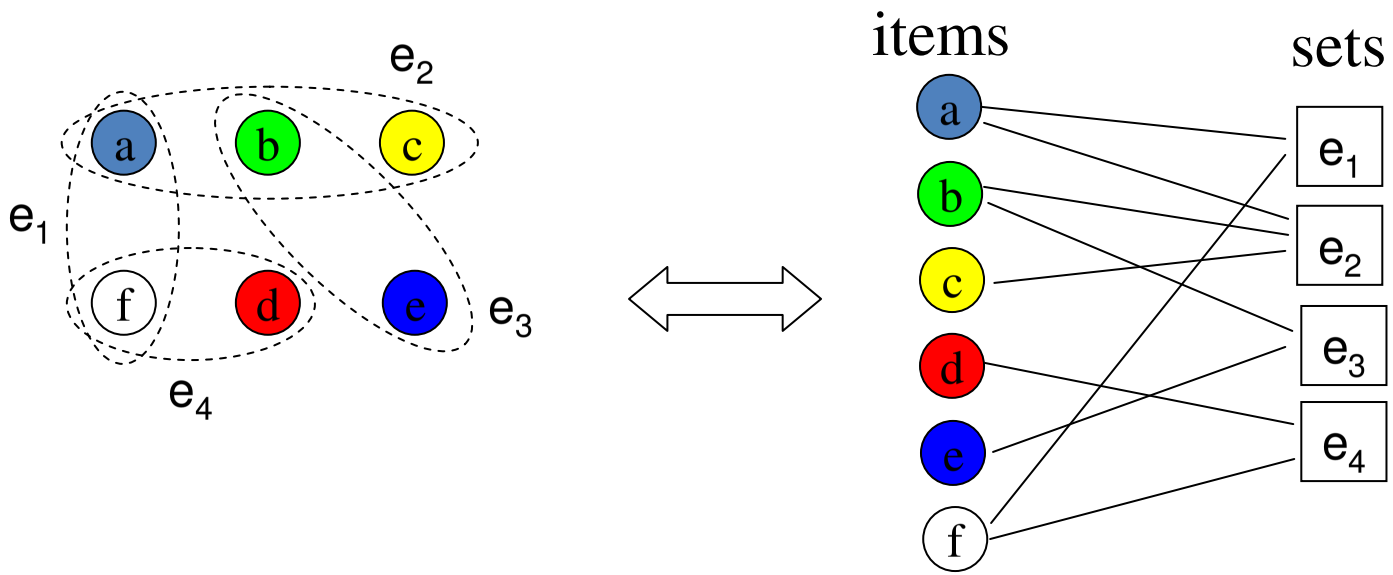
Optimal vs. greedy histogram



- Shrink height and width of ALG by factor 2
- Align to the right

- suppose *p* is high ➔
- weight covered (δ) is small w.r.t. remaining weight (W) ➔
- no algorithm can clear more than δ in any following step ➔
- OPT must make at least (W-W/2) / δ steps to be left with only W/2 remaining weight.

Cover time: Last vertex in Hyperedge



– this is the min latency variant of set cover: order items to minimize sum of complete covering times of sets.

Algorithm for Min Latency

- Notation:
  - x_v is the rank of vertex v
  - y_e is the cover time of e

$$\text{minimize} \sum_{e \in E} w(e) y_e$$

$$\text{subject to } (1) \;\; y_e \geq x_v \qquad \forall v \in e$$

$$(3) \;\; y_e \in \mathbf{Z}_+, \; x_v \in \pi_{(1,2,\ldots n)}$$

LP relaxation

$$\text{minimize} \sum_{e \in E} w(e)\, y_e$$

$$\text{subject to (1)} \quad y_e \geq x_v \qquad \forall v \in e$$

$$\text{(2)} \quad \sum_{v \in S} x_v \geq (|S|^2 + |S|)/2 \qquad \forall S \subseteq V$$

$$\text{(3)} \quad y_e \geq 0,\ x_v \geq 0$$

$$\text{minimize} \sum_{e \in E} w(e)\, y_e$$

$$\text{subject to (1)} \quad y_e \geq x_v \qquad \forall v \in e$$

$$\text{(3)} \quad y_e \in \mathbf{Z}_+,\ x_v \in \pi_{(1,2,\ldots n)}$$

How to solve the LP relaxation?

- We need a poly time oracle to check if a solution is feasible

$$
\begin{aligned}
\text{minimize} \quad & \sum_{e \in E} w(e)\, y_e \\
\text{subject to} \quad & (1) \quad y_e \geq x_v && \forall v \in e \\
& (2) \quad \sum_{v \in S} x_v \geq (|S|^2 + |S|)/2 && \forall S \subseteq V \\
& (3) \quad y_e \geq 0,\ x_v \geq 0
\end{aligned}
$$

- Given a solution
  - checking (1) and (3) is immediate
  - for (2):

    what is the S of size 1 with the minimum sum ?

    what is the S of size 2 with the minimum sum ?

    what is the S of size 3 with the minimum sum ?

How to round the solution?

- x_v needs to be a permutation of (1,2,…n)
- Sort them according to value
- This will give the permutation
- y_e will be the maximum of x_v for all v in e

$$
\begin{aligned}
\text{minimize} \quad & \sum_{e \in E} w(e)\, y_e \\
\text{subject to} \quad & (1) \quad y_e \geq x_v && \forall v \in e \\
& (2) \quad \sum_{v \in S} x_v \geq (|S|^2 + |S|)/2 && \forall S \subseteq V \\
& (3) \quad y_e \geq 0,\ x_v \geq 0
\end{aligned}
$$

What is the integrality gap?

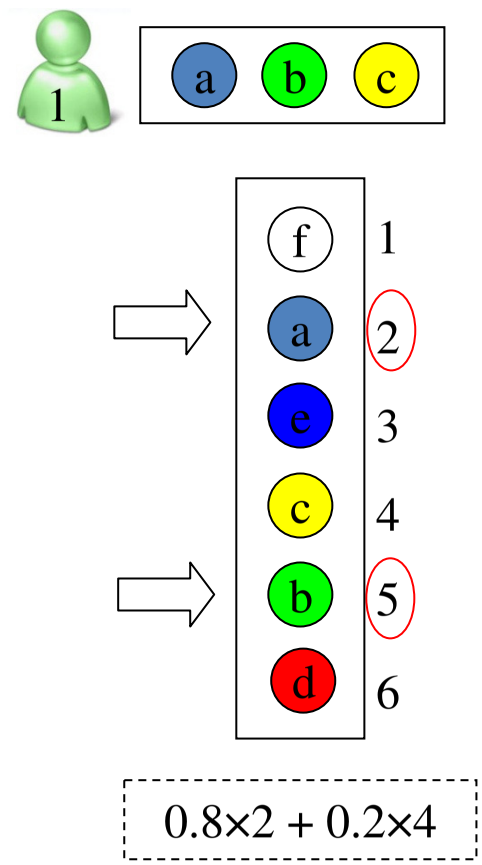
- Let a1, a2,… be the sorted values of the x's
- $a_1 \geq 1$
- $a_1+a_2 \geq 1+2$ ➔ $a_2 \geq (1+2)/2$ since $a_2 \geq a_1$
- ….
- $a_1+a_2+\ldots+a_k \geq 1+2+\ldots+k$ ➔ $a_k \geq (1+2+\ldots+k)/k$
- Hence, $a_k \geq (k+1)/2$… but it was rounded to k

What we got here

Min Latency: 2-approximation

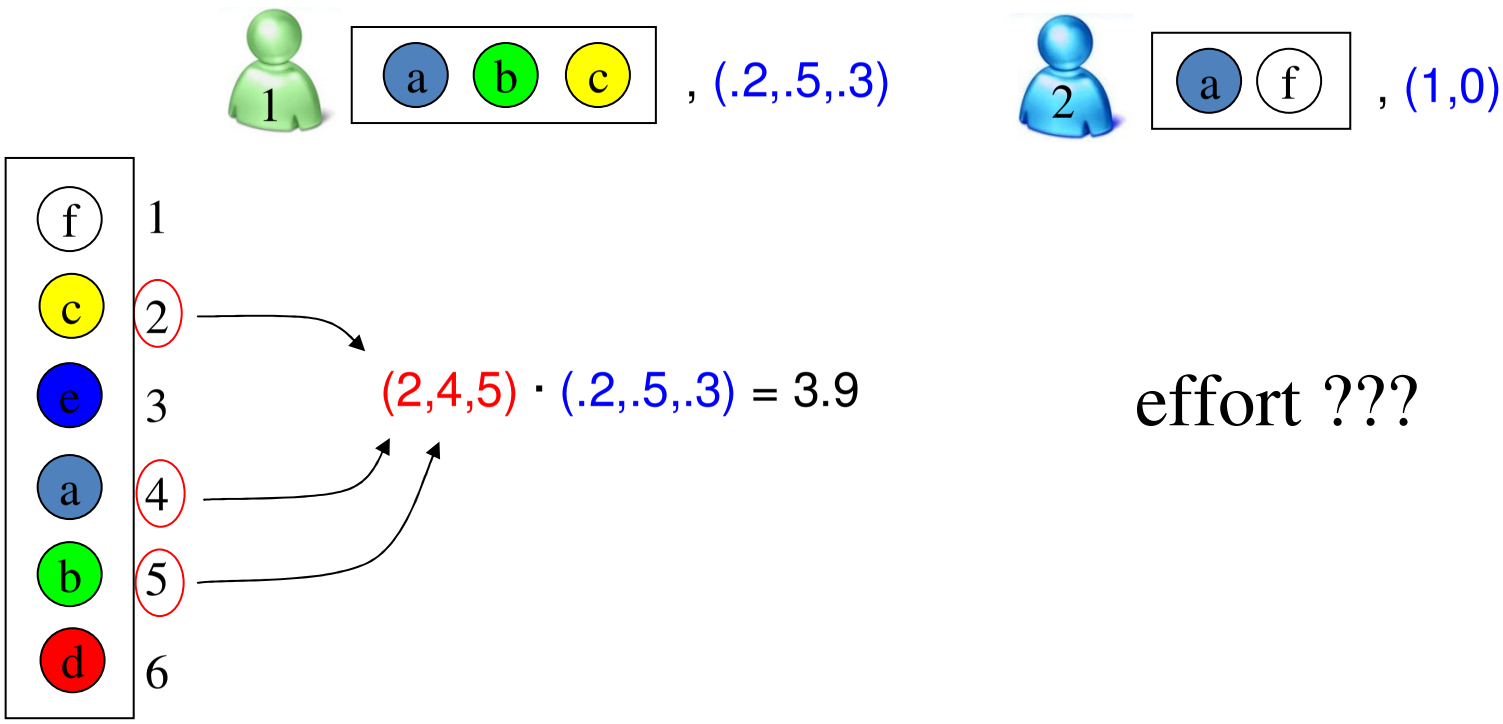
Reminder: user effort

- What is the effort of a user type?
  - looking for first relevant result? (4-approx)
  - looking for all relevant results? (2-approx)
  - maybe user type is more complex? e.g., must see first relevant result but with 20% also the second one?
- In practice, all options exists…



Profile vectors

- A user (hyperedge) also has a profile vector:



profile vector weights correspond to positions and not items!

More on profile vectors

- Intention of user types:
  - looking for first relevant result? (1, 0, …, 0)
  - looking for all relevant results? (0, …, 0, 1)
  - looking for first relevant result but with 20% also the second one? (0.8, 0.2, 0, …, 0)
  - non-increasing profile
  - non-decreasing profile

- Proportion of user types:
  - sum of entries ≈ influence.

(1,0,…,0)
vs.
(200,0,…,0)

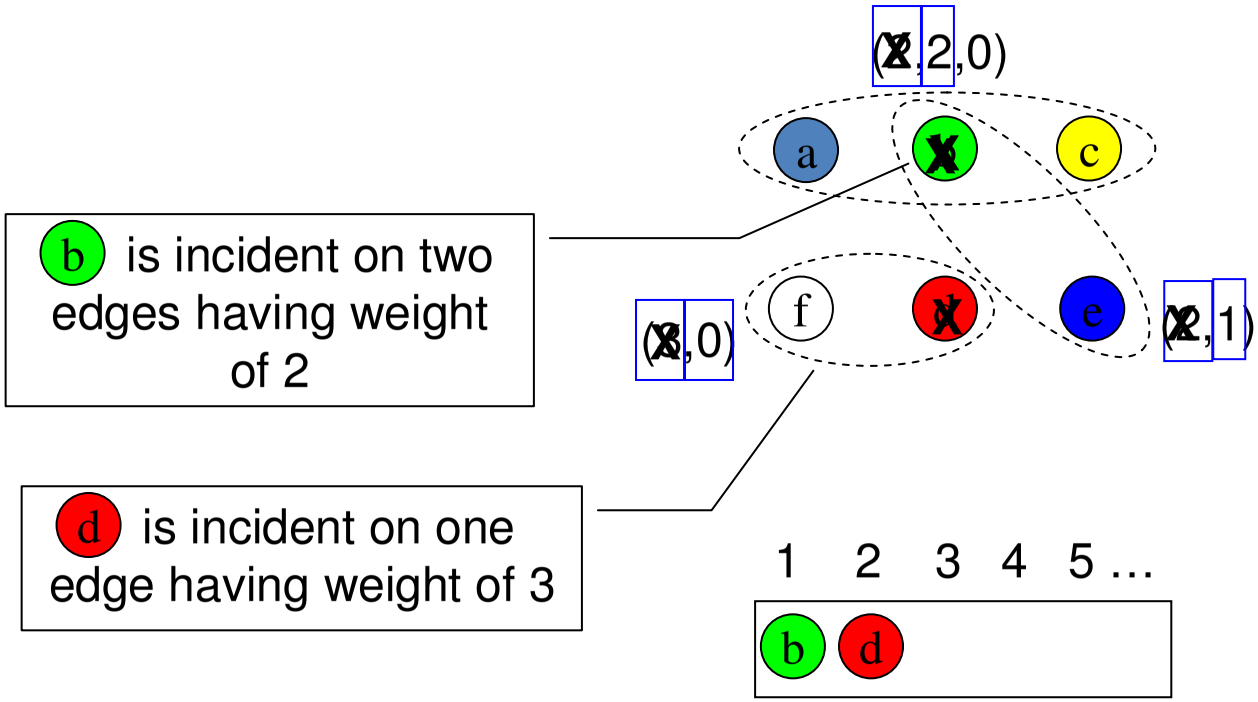
Non-increasing profile

Non-increasing profiles: 4-approximation

- generalizes min-sum set cover - (1,0,…,0).
- based on a weight reduction greedy algorithm

Weight reduction greedy algorithm

Greedily select a vertex that maximizes the overall weight of incident hyperedges



Weight reduction greedy algorithm

- Remarks:
  - min-sum set cover (1,0,…,0) - the algorithm reduces to the natural greedy algorithm of set cover:

    select a vertex that covers a maximal number of un-covered hyperedges.

  - the algorithm is not the only possible generalization of the greedy algorithm of set cover.

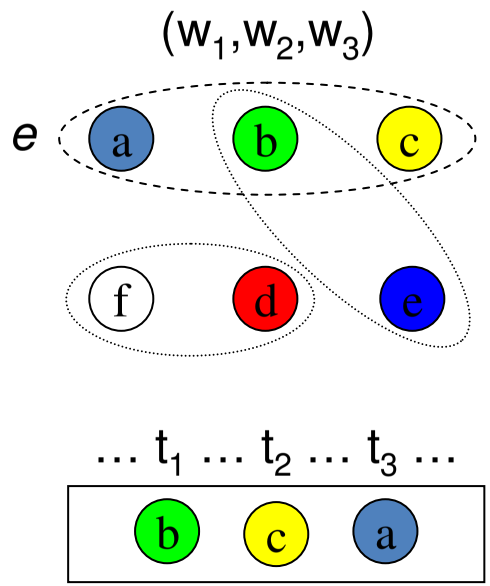
(2,2,0) → (2,2,0) - consider the next coordinate

(2,2,0) → (2,2,0) - consider the sum of remaining coordinates

Weight reduction greedy algorithm

- Motivation:
  - the covering weight of *e* is $w_1t_1 + w_2t_2 + w_3t_3$.
  - consider the ordering as time:

    time $[0, t_1)$ cost $= w_1 + w_2 + w_3$
    time $[t_1, t_2)$ cost $= w_2 + w_3$
    time $[t_2, t_3)$ cost $= w_3$



⇨ selecting one vertex of *e* results in a reduction of w_1 for remaining time…

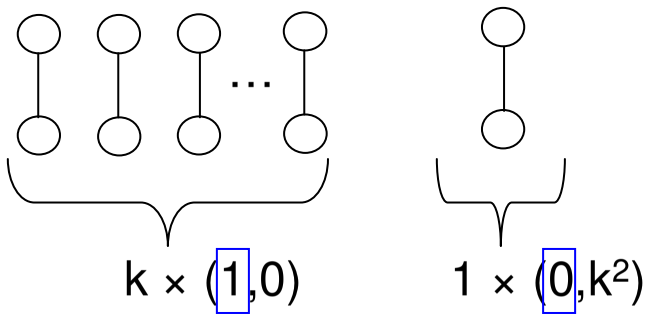
Weight reduction greedy algorithm

- Weight reduction greedy is 4-approxmation for non-increasing weight vectors.

Does weight reduction greedy work for arbitrary weight vectors?

Greedy approach fails in general

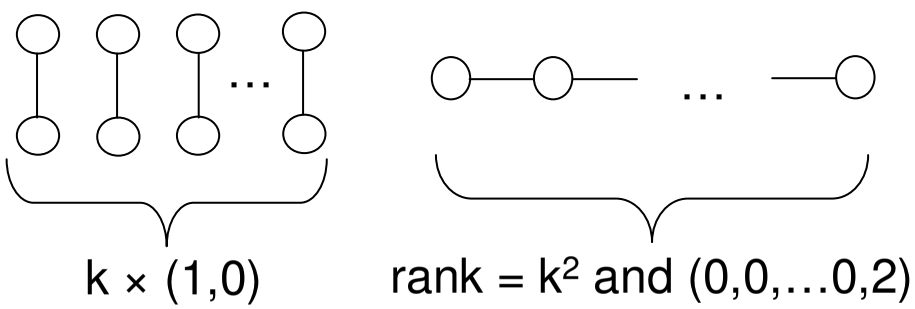
– greedy fails to look beyond the next step…



ALG ≈ k^3

OPT ≈ k^2

- tweak greedy to consider sum of remaining weights…



ALG ≈ k^3

OPT ≈ k^2

Harmonic interpolation algorithm

- Spread weights in a non-uniform way.

$$w(e) = (w_1, \ldots, w_r) \quad \Rightarrow \quad \tilde{w}_i(e) = \sum_{j=i}^{r} \frac{w_j}{j-i+1}$$

- Run weight reduction greedy with respect to $\tilde{w}$.

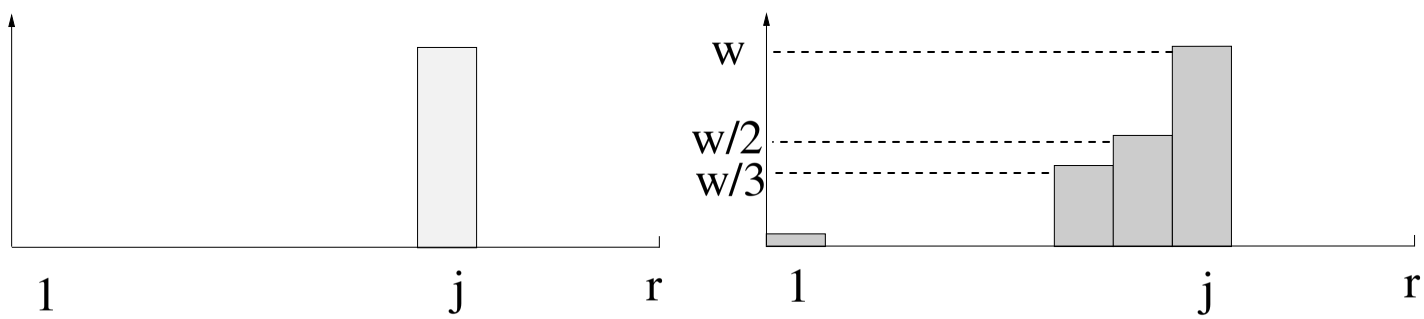
Harmonic interpolation

General profiles: $O(\log r)$-approximation

– r is the maximal rank of a hyperedge.

Harmonic interpolation

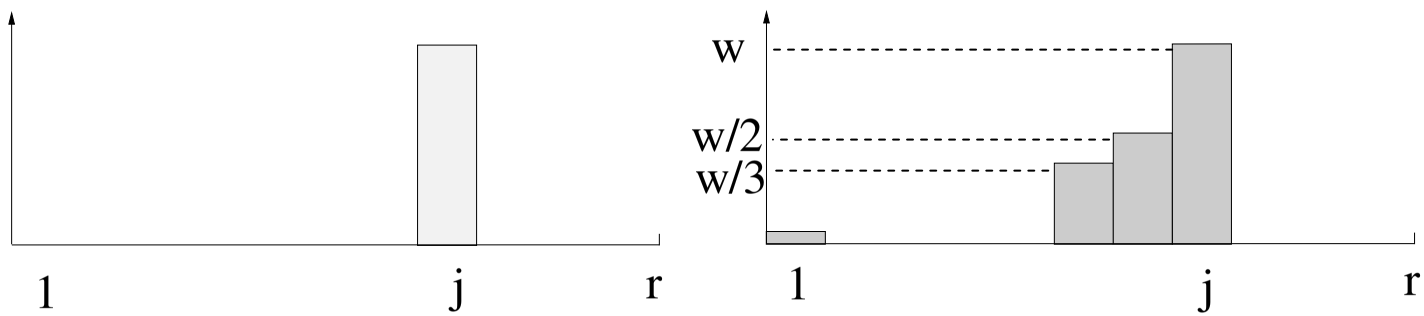
- Consider an indicator vector (0, …, 0, w, 0, …, 0) resulting harmonic vector is $(\frac{w}{j}, \ldots, \frac{w}{2}, \underset{\substack{\uparrow \\ j}}{w}, 0, \ldots, 0)$



  – Intuitively, greedy gains some knowledge about future weight reduction from any coordinate.

The arbitrary case analysis

- The original analysis does not hold both before and after the harmonic interpolation.



  - Intuitively, the idea is to reduce the weights and make them non-increasing in a dynamic fashion.

Concluding remarks

- [A, Gamzu, Yin]
  - model for re-ranking.
  - general profiles: $O(\log r)$-approximation.
  - non-increasing profiles: 4-approximation.
  - non-decreasing profiles: 2-approximation.
- Open questions:
  - does $O(\log r)$-approximation is best possible?
  - find additional applications for harmonic interpolation.

Thank You!