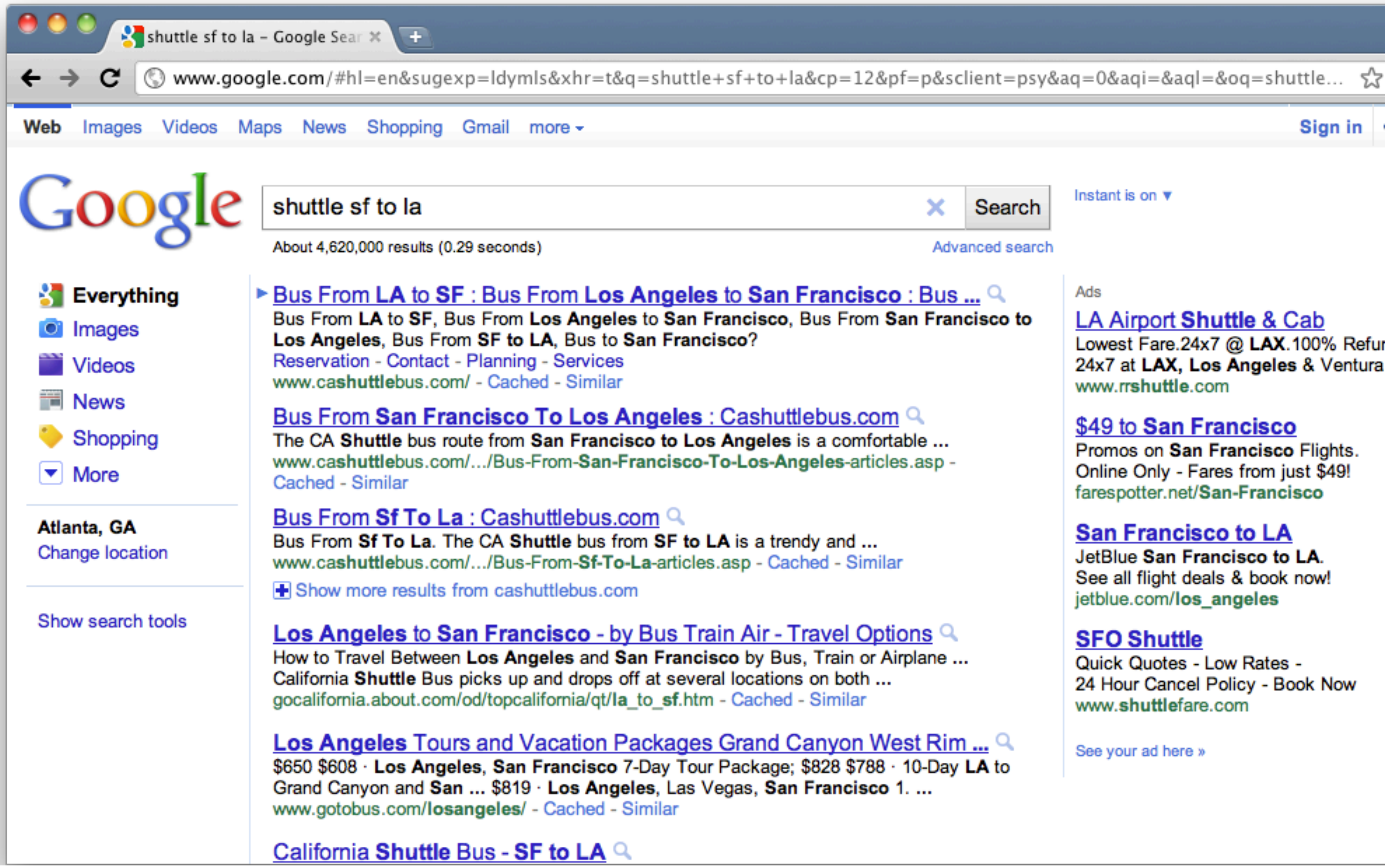
Online Matching and Adwords

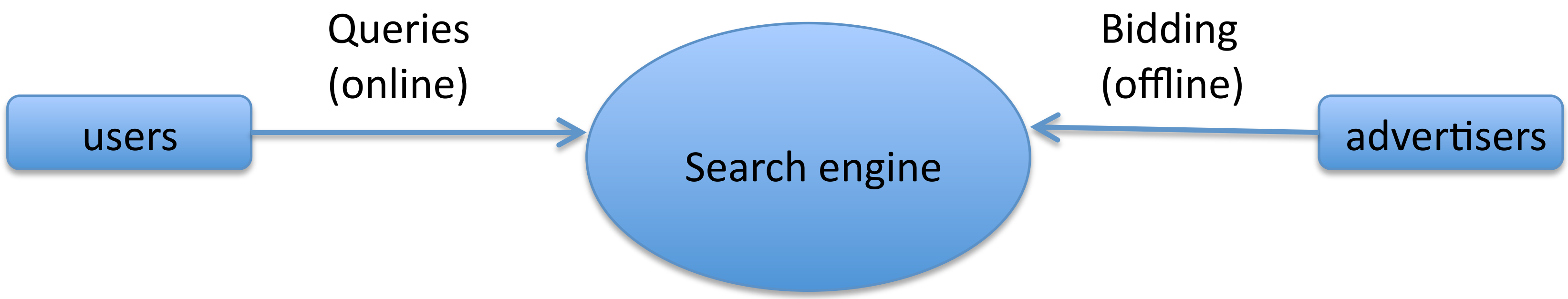
Aranyak Mehta

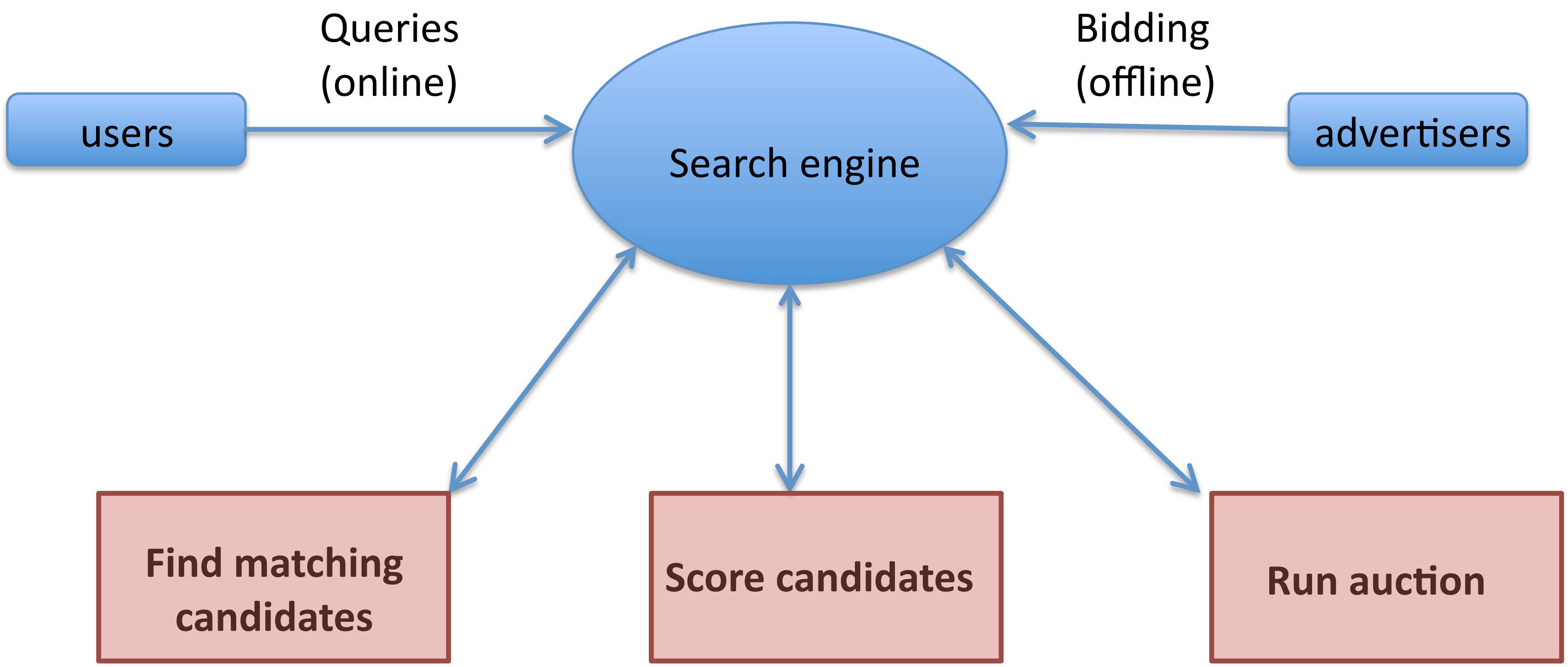
Google Research

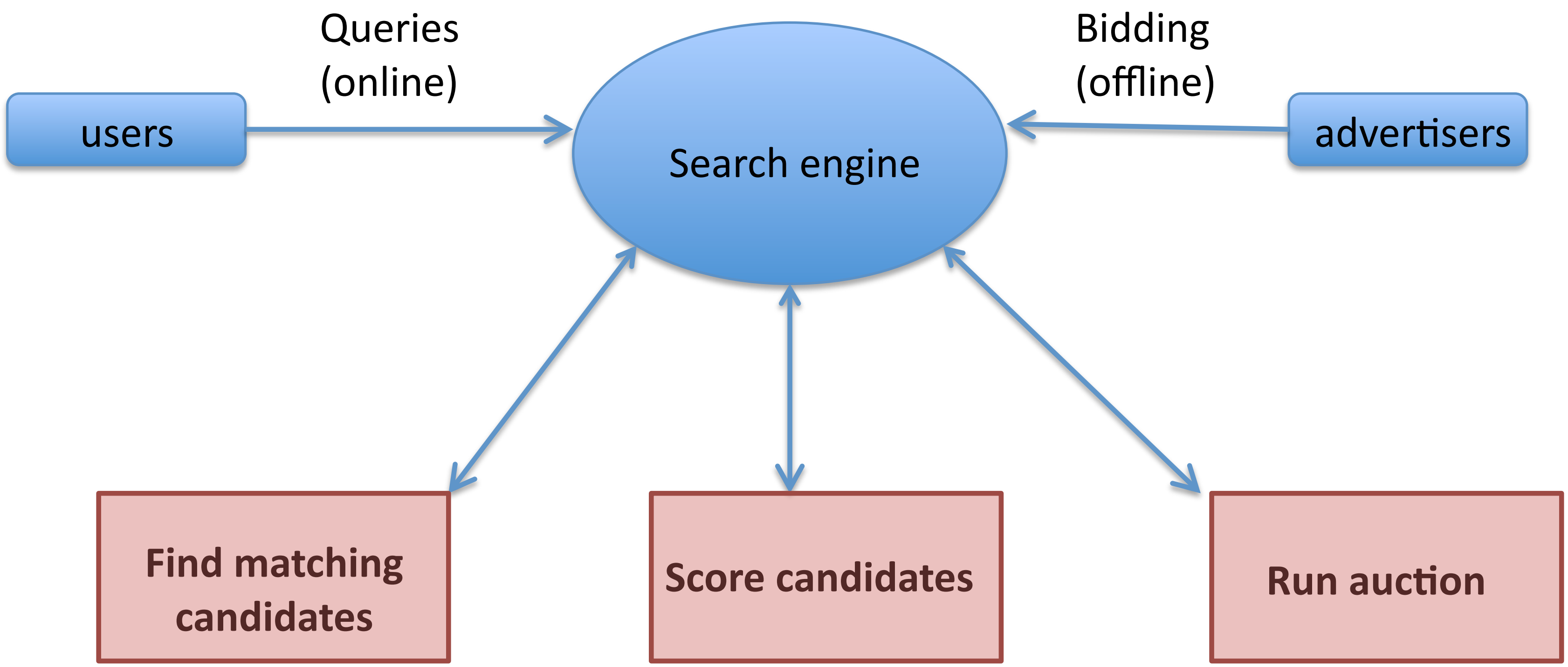
Mountain View, CA

Search Advertising (Adwords)



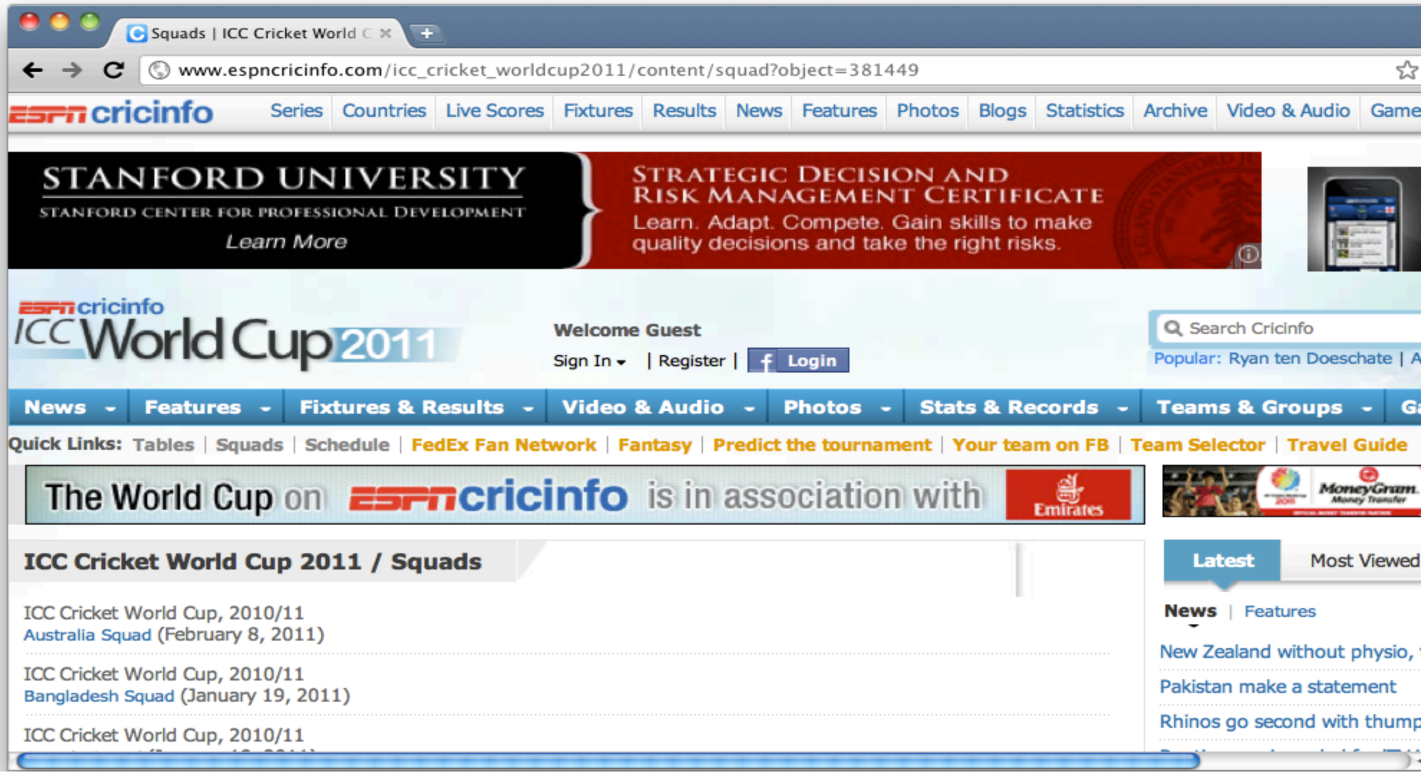

users
Queries
(online)
Search engine
Bidding
(offline)
advertisers


Queries
(online)
Bidding
(offline)
users
Search engine
advertisers
Find matching candidates
Score candidates
Run auction



- Advertiser Budgets => Demand Constraint => Matching

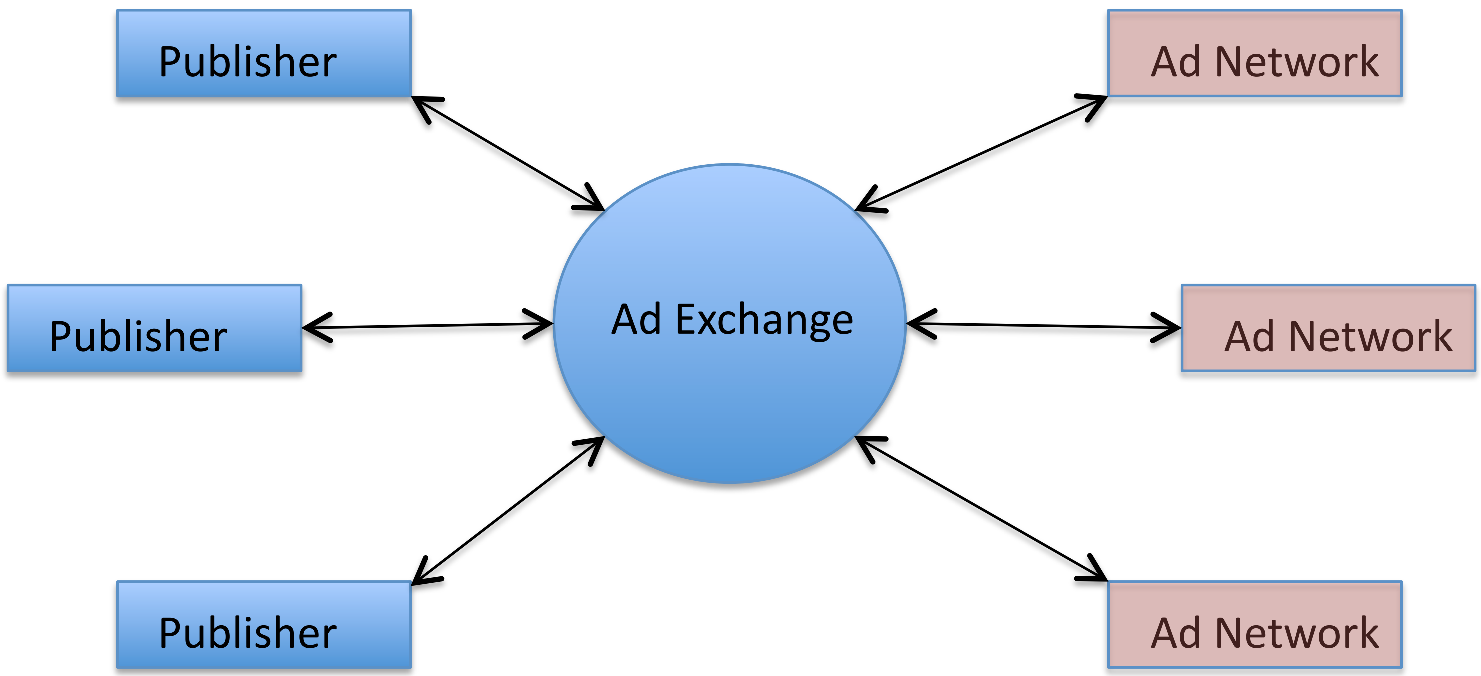
Display ads



- Demand is offline from advertisers
  - Targeting
  - Quantity ("1M ads")

- Supply is online from page views

Ad Exchanges

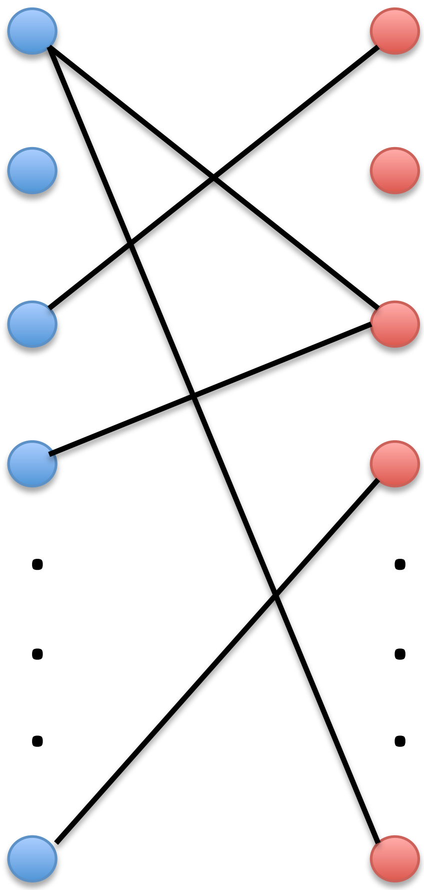


… Many other examples from advertising and outside.

Theory to Practice

- Details of the implementation:
  - CTR, CPC, pay-per-click, Second price auction, Position Normalizers.
- Objective functions (in order):
  - User's utility
  - Advertiser ROI
  - Short term Revenue / Long term growth

Online Bipartite Matching

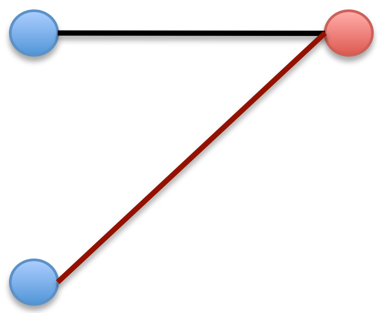


Known in advance

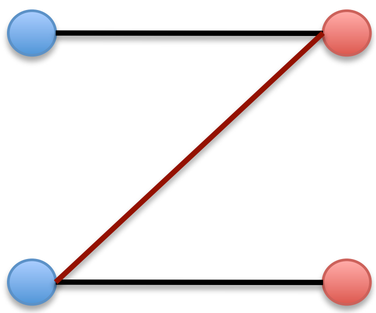
Arrive Online

- Match upon arrival, **irrevocably**
- Maximize size of the matching

The Core difficulty

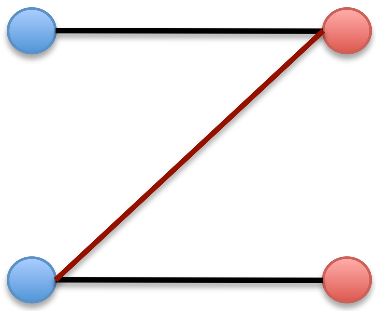


The Core difficulty

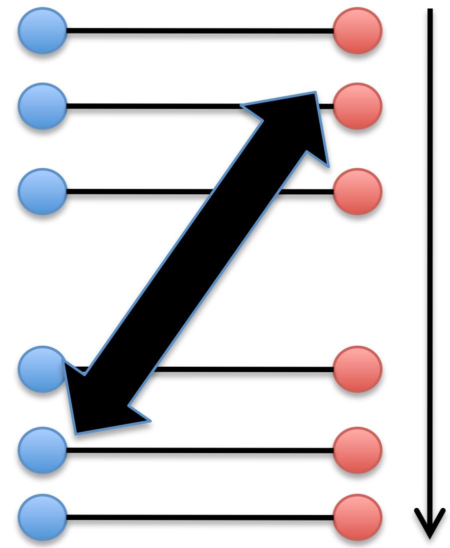


No deterministic algorithm can do better than 1/2

The Core difficulty



No deterministic algorithm can do better than 1/2



RANDOM is 1/2

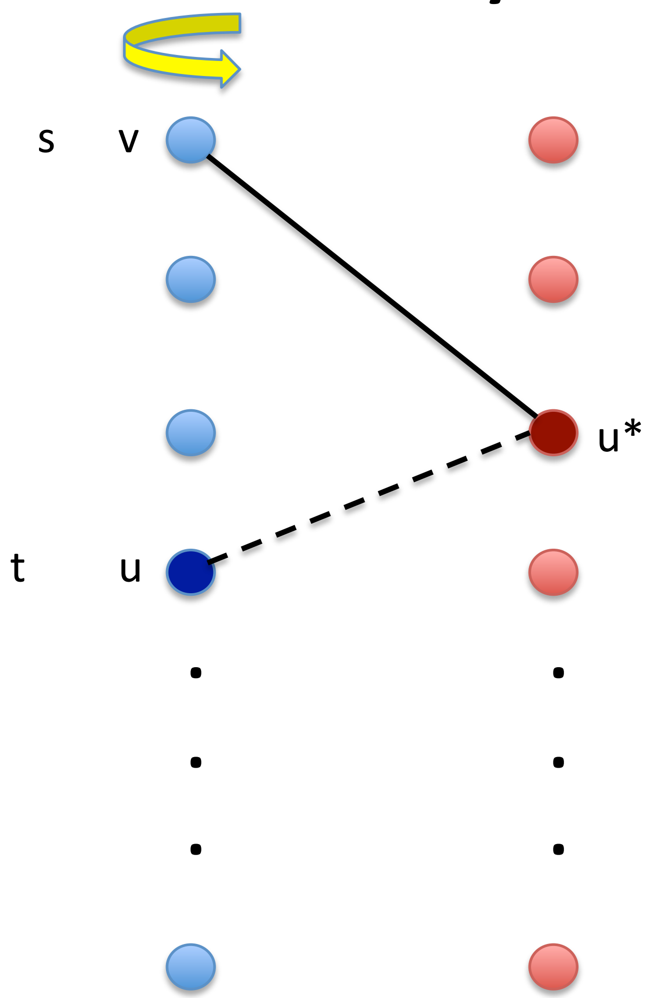
KVV: Correlated Randomness

[Karp, Vazirani and Vazirani STOC 1990]

- Randomly permute vertices in L
- When a vertex in R arrives:
  - Match it to the highest available neighbor in L

Theorem: KVV achieves a factor of 1-1/e.
This is optimal.

Analysis of KVV in 1 slide



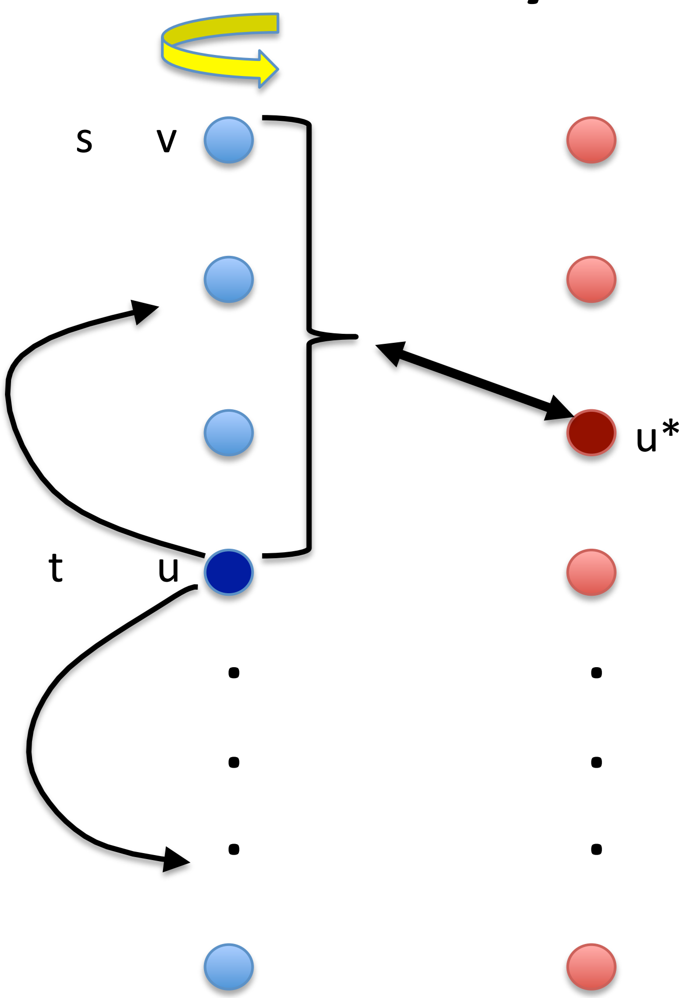
[Birnbaum, Matheiu] SIGACT News 2008

u miss @ t

⇒ u* match @ $s < t$

⇒ Factor $\geq$ 1/2

Analysis of KVV in 1 slide

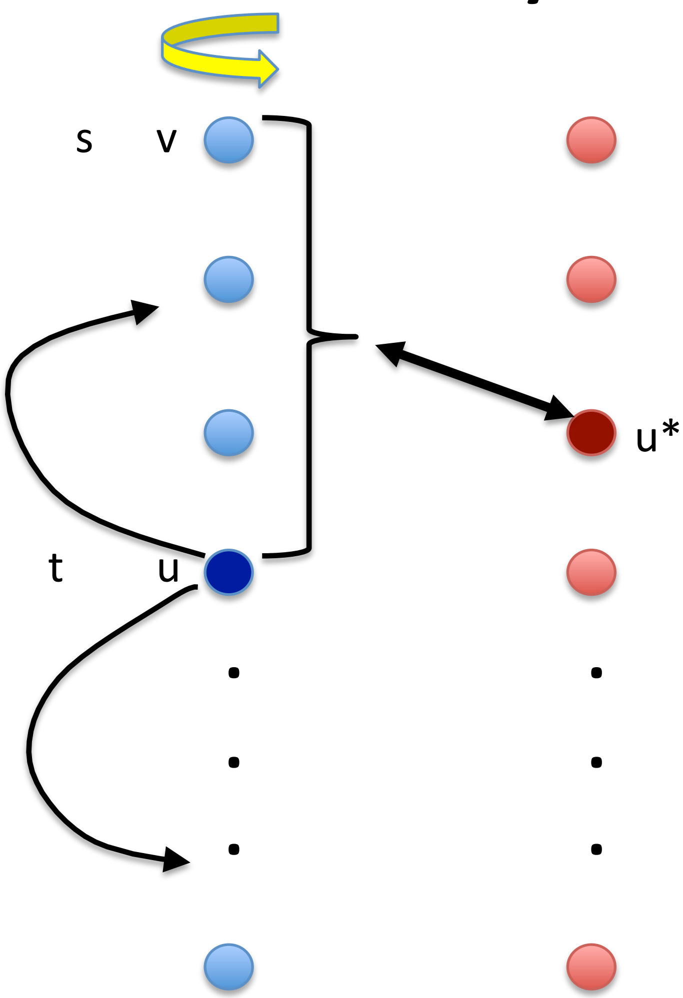


[Birnbaum, Matheiu] SIGACT News 2008

u miss @ t

⇒ In each of the n permutations
u* match at s ≤ t

Analysis of KVV in 1 slide



[Birnbaum, Matheiu] SIGACT News 2008

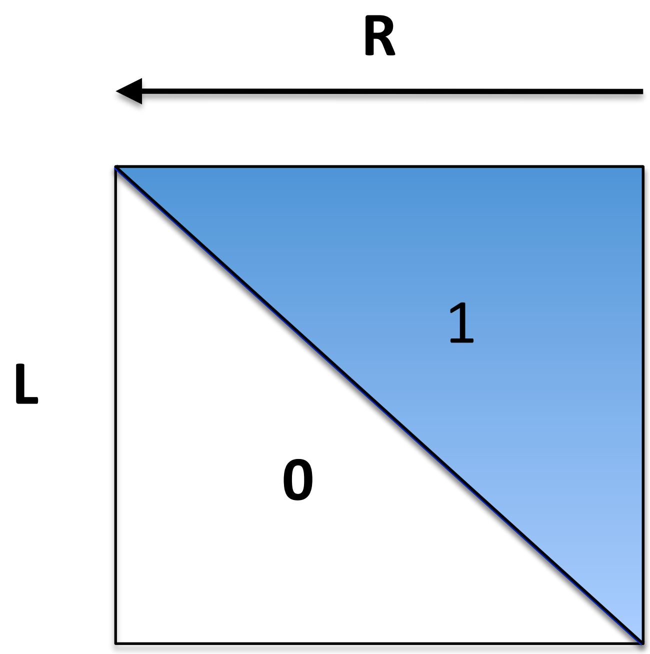
u miss @ t

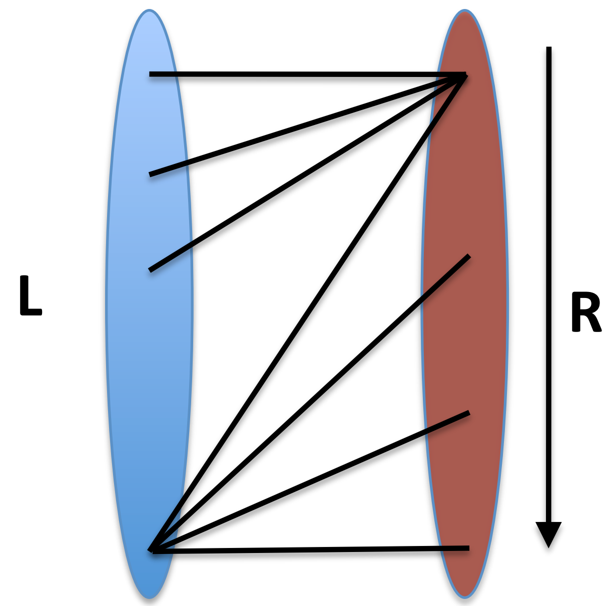
⇒ In each of the n permutations
u* match at s ≤ t

⇒ $\Pr[miss@t] \leq \frac{1}{n} \sum_{s \leq t} \Pr[match@s]$

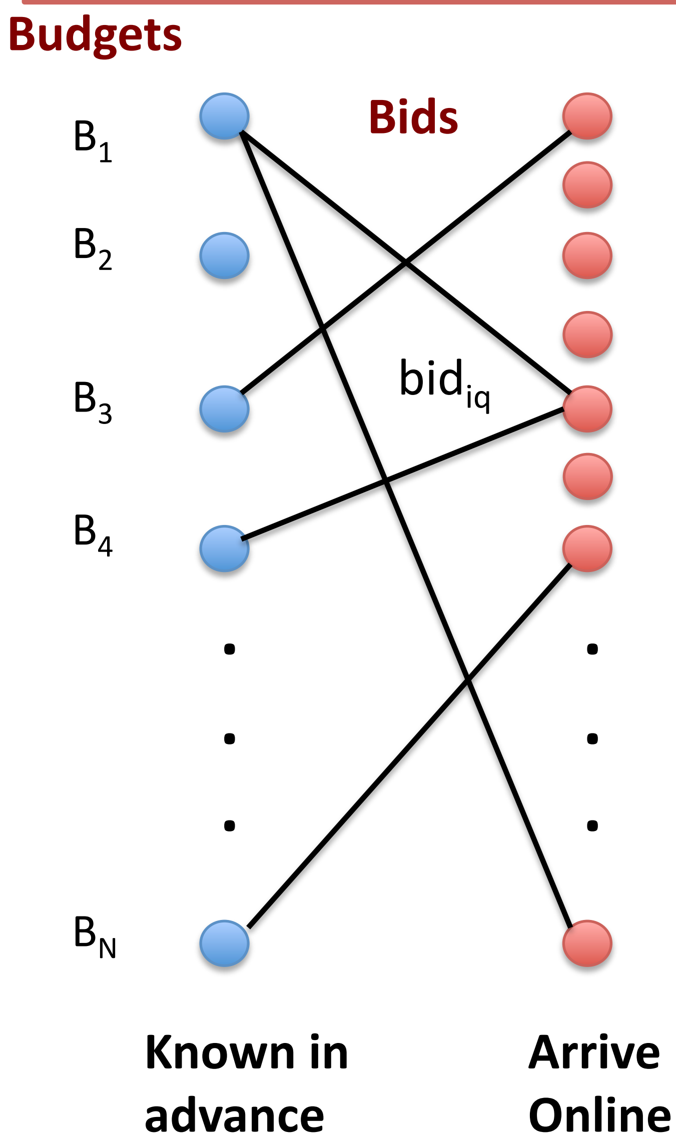
⇒ Factor 1-1/e

Tight example





Goal: "Adwords" problem



- **bid << budget**
- Maximize the sum of budgets spent

4 models of arrival

Adversarial.

Random Order: The set of vertices is adversarial, but arrive in a random order.

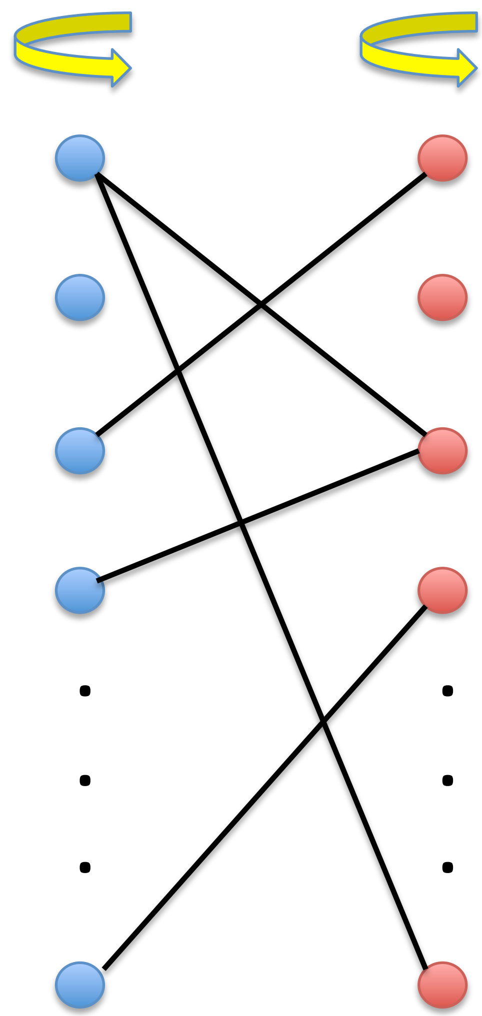
Unknown Distribution: Each vertex is picked iid from some (unknown) distribution.

Known Distribution: Each vertex is picked iid from a known distribution.

	Adversarial Order	Random order / Unknown iid	Known iid
Bipartite Matching	1-1/e (optimal)		
Vertex Weighted Matching			
Adwords			

Unknown Distribution / Random Order

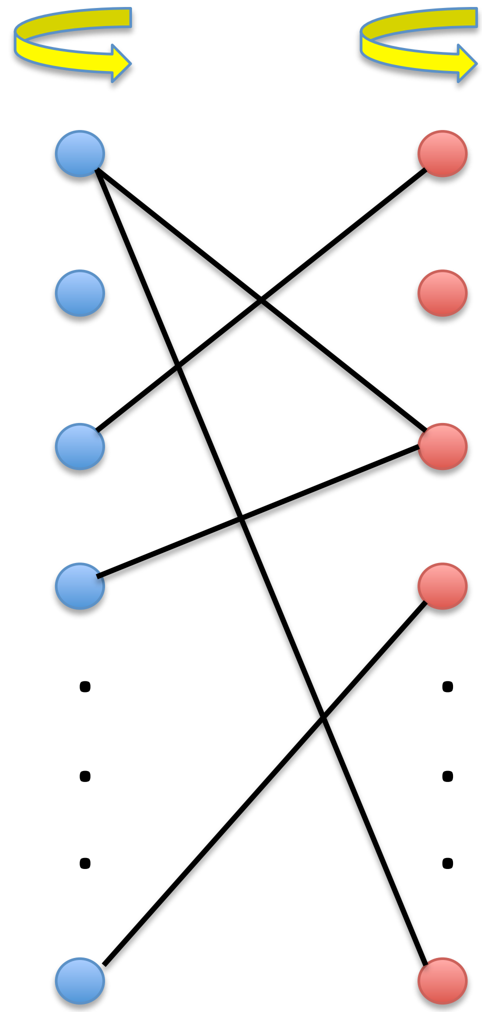
- GREEDY = 1-1/e
- KVV >= 1-1/e
- Can we do better?
  How about KVV itself?



Unknown Distribution / Random Order

- GREEDY = 1-1/e
- KVV >= 1-1/e
- Can we do better?
  How about KVV itself?

Upper triangular example
goes to factor 1 !



KVV in random order

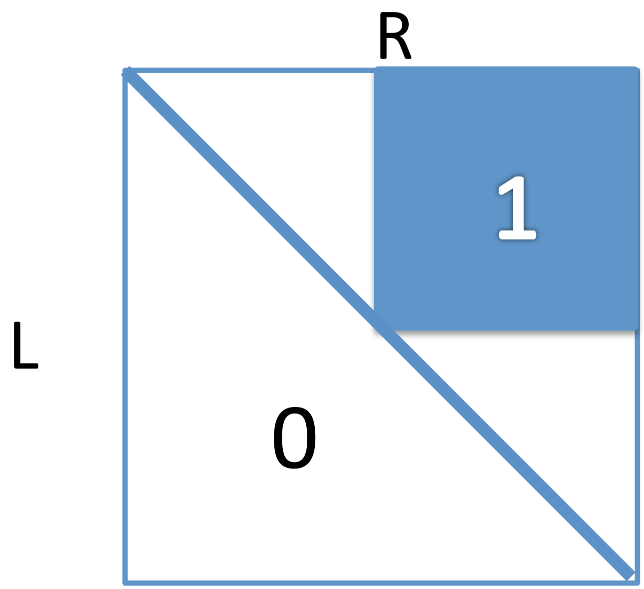
[Karande, Mehta, Tripathi] STOC 2011

Theorem 1: KVV has factor 0.655 in the Random Order Model
[Computer aided proof: 0.667]

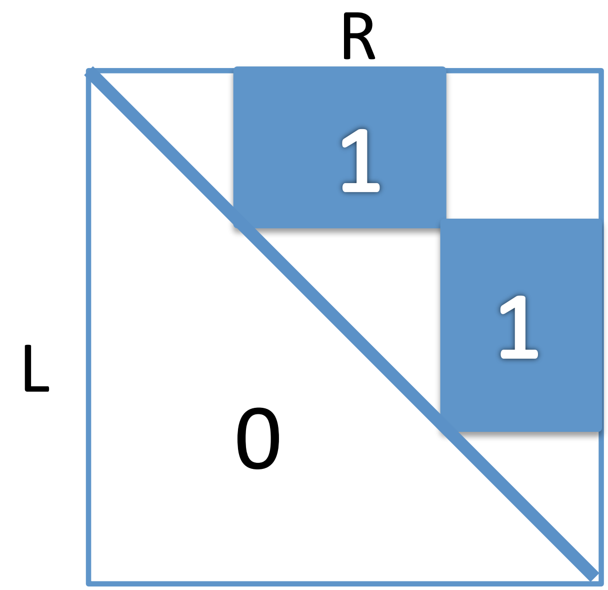
Theorem 2: : If the graph has k disjoint perfect matchings then factor is
$1 - 1/\sqrt{k}$

[Mahdian, Yan] STOC 2011 Computer aided proof for 0.696

Bad examples



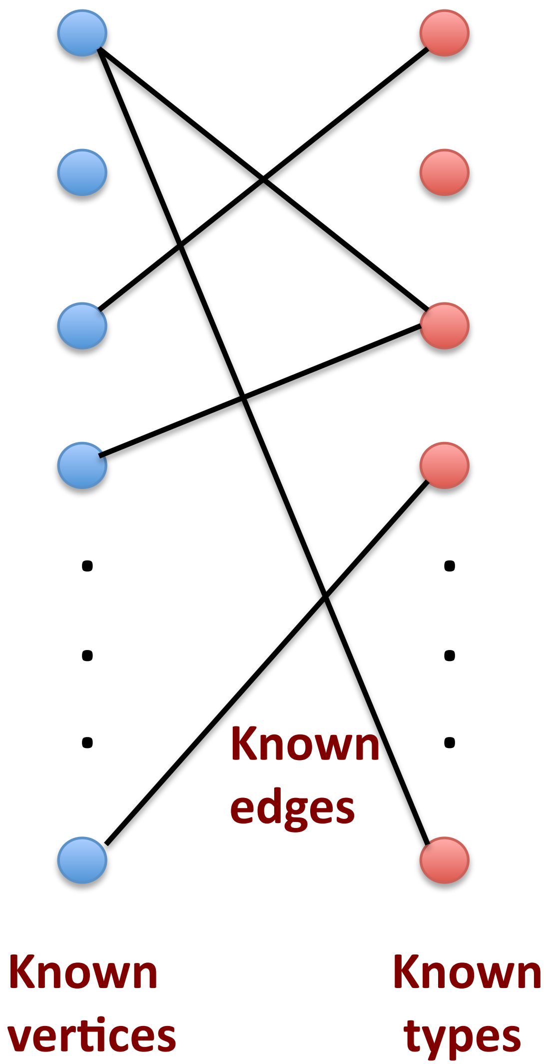
0.75



0.726

	Adverserial Order	Random order / Unknown iid	Known iid
Bipartite Matching	1-1/e (optimal)	0.696 (0.83)	
Vertex Weighted Matching			
Adwords			

Known iid input

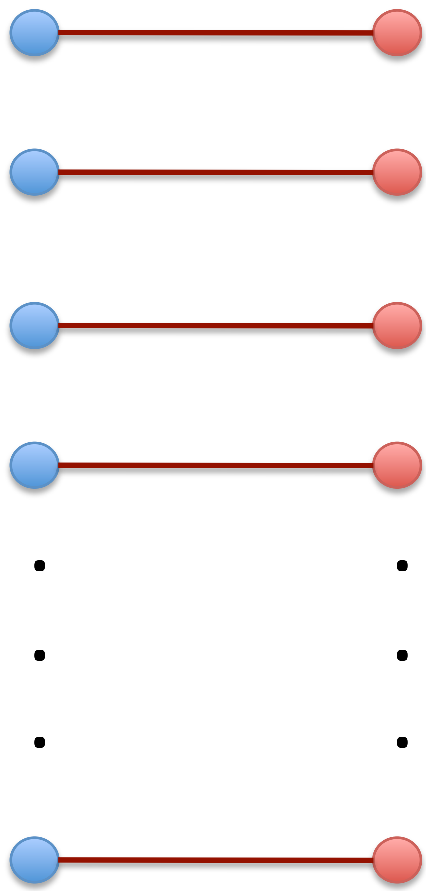


[Feldman, Mehta, Mirrokni, Muthukrishnan] FOCS 2009

Vertices in R are picked iid from the set of types (with replacement)

Use offline estimates to guide online decisions?

Attempt 1



Optimal matching in Base Graph

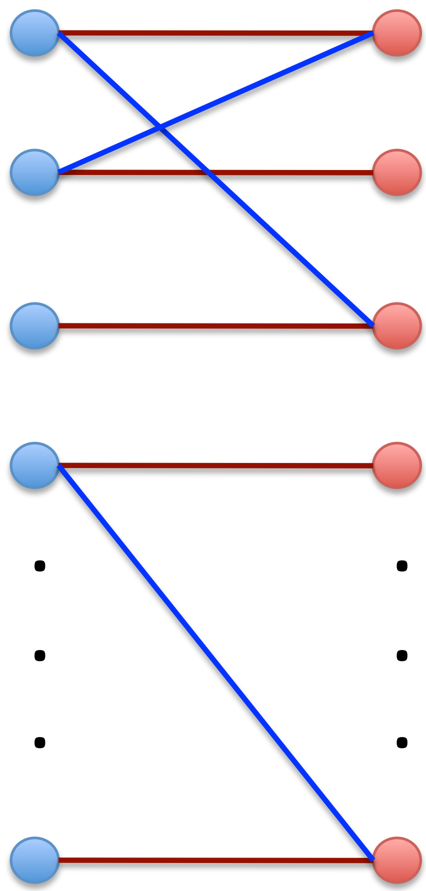
ALGORITHM Suggested-Matching:

- Find an optimal matching in the base graph
- When the next vertex arrives:
  - If the optimal match is available, use it
  - Else don't match

Core difficulty:
Some types will repeat.
You will match only the first of each type.

$\Rightarrow$ **Factor 1-1/e**

Attempt 2: Power of two choices!



Two Matchings
in Base Graph

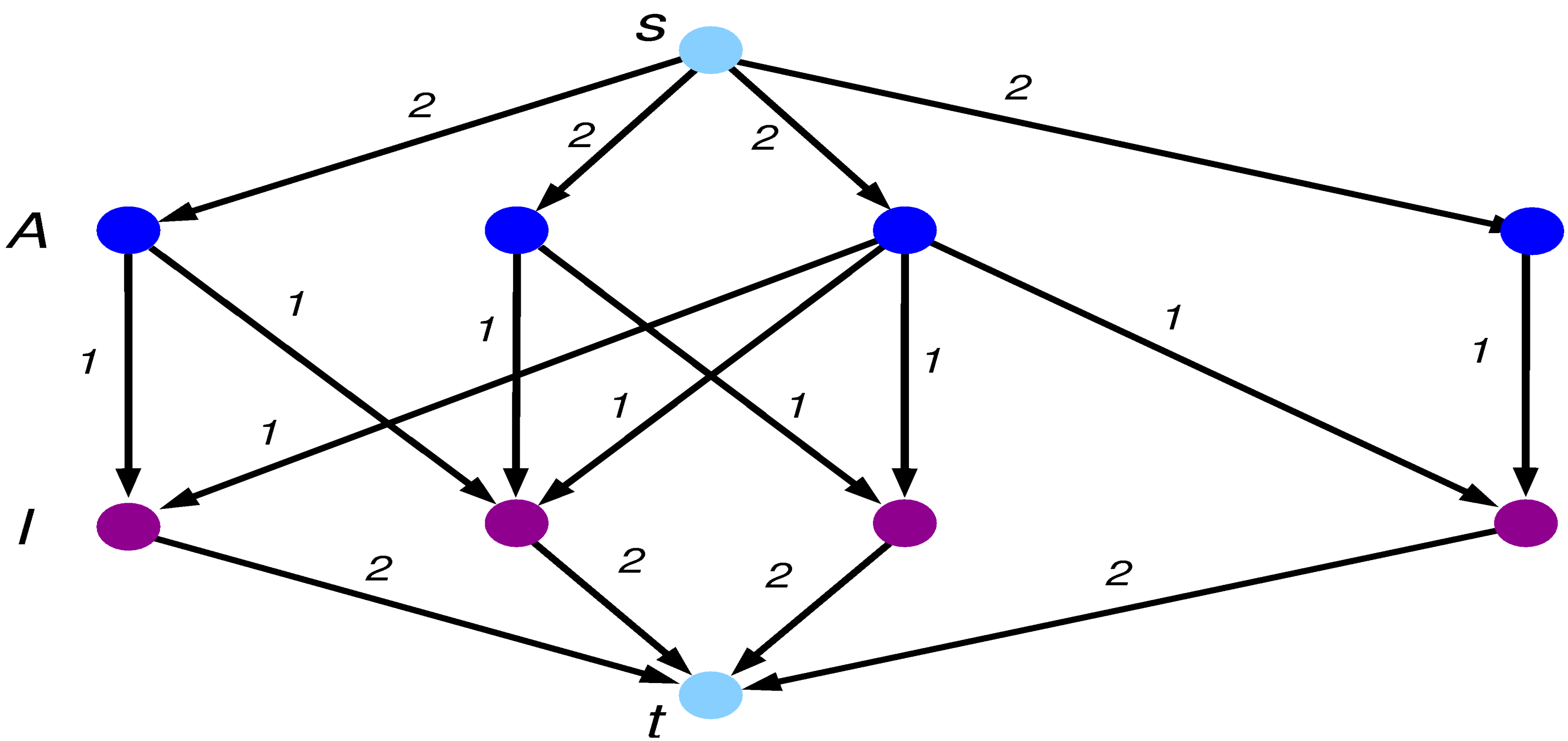
ALGORITHM Two-Suggested-Matchings (TSM):

Offline: Find Two disjoint matchings in base graph

Online:

- **Try Red Matching**
  - **If FAIL : Try Blue Matching**
    - **If FAIL: do not match**

How to get the two matchings?



Performance

- Theorem: TSM achieves factor 0.67 with high probability

$$\frac{1-\frac{2}{e^2}}{\frac{4}{3}-\frac{2}{3e}}$$

  - Miraculously, this is tight!
  - No algorithm can get 1- o(1)

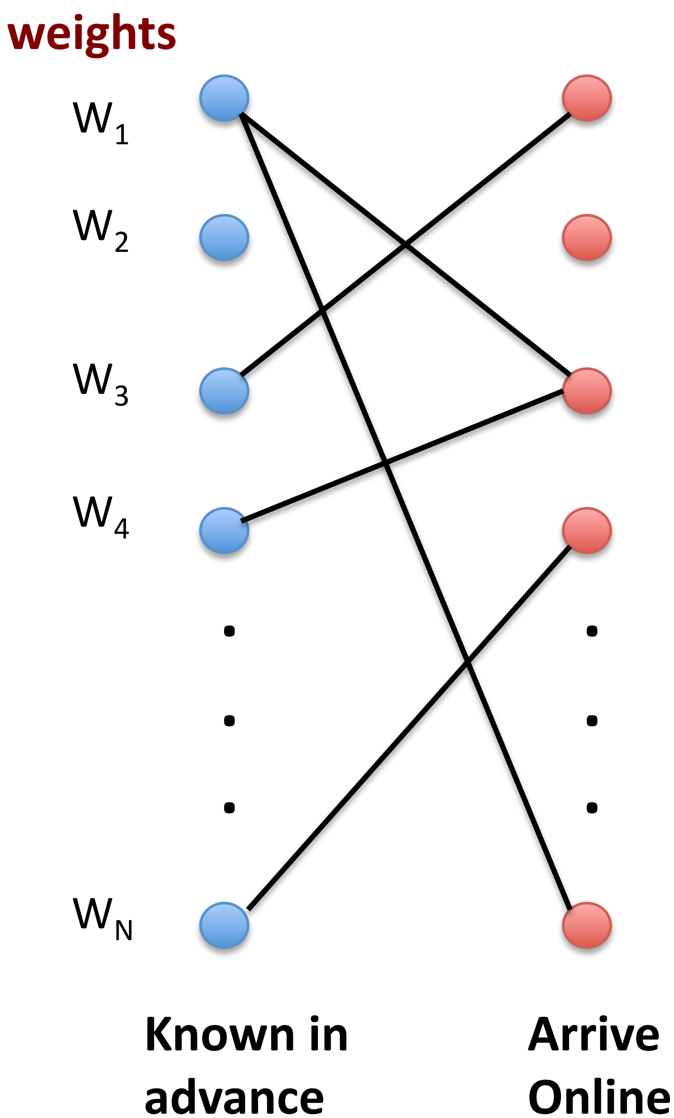
Follow-ups:

[Bahmani Karpalov] (ESA 2010): 0.70 and upper bound of 0.90

[Manshadi, Oweis-Gharan, Saberi] (SODA 2011): 0.70 and upper bound of 0.83

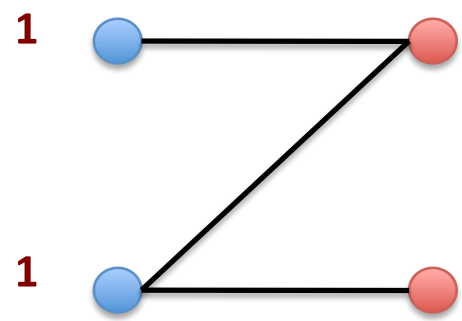
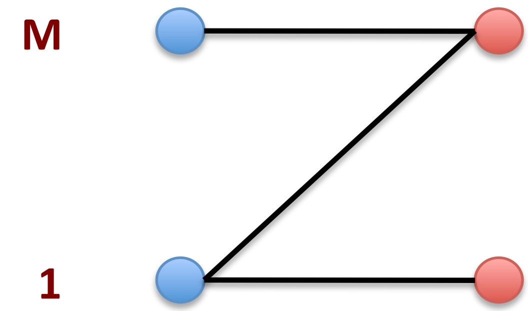
	Adverserial Order	Random order / Unknown iid	Known iid
Bipartite Matching	1-1/e (optimal)	0.655 (0.83)	0.70 (0.83)
Vertex Weighted Matching			
Adwords			

Weighted Vertex Matching

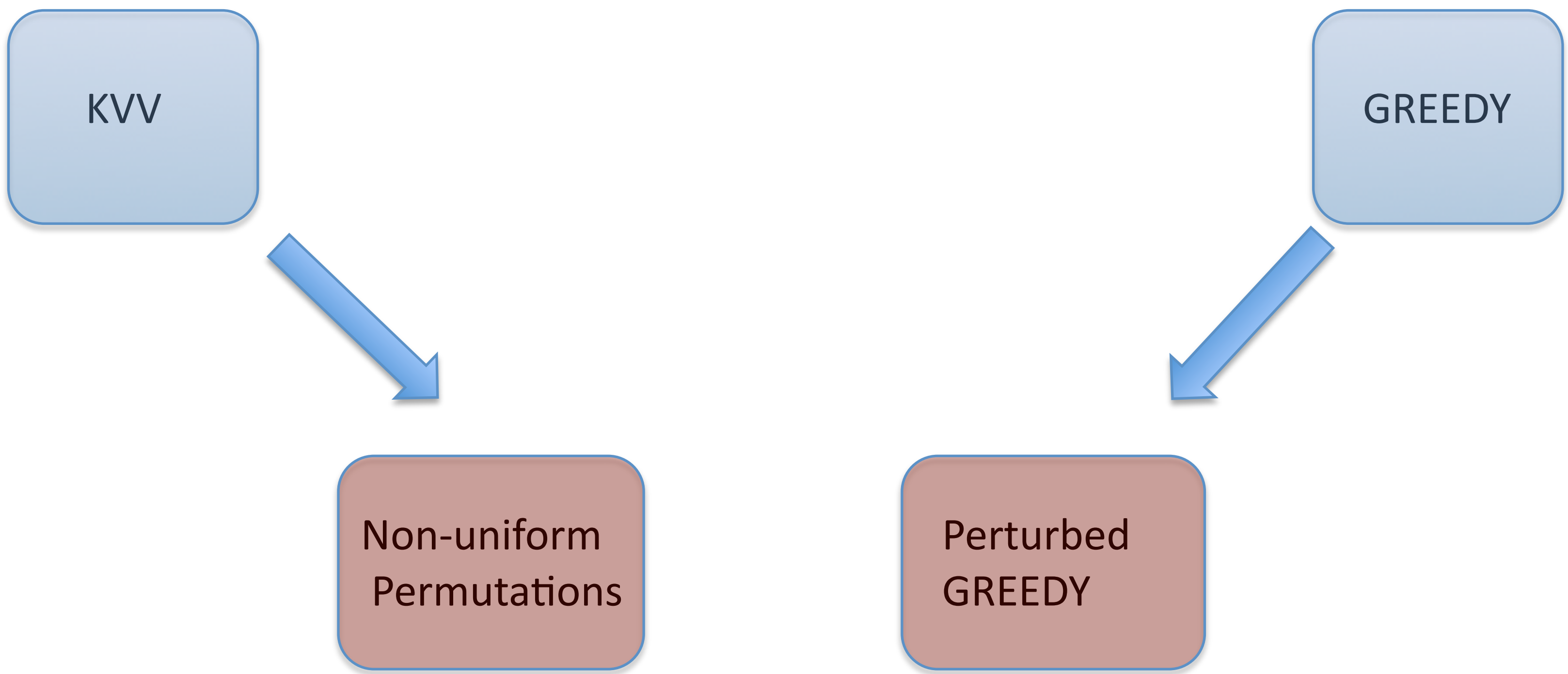


- Vertices in L have weights
- Vertices in R arrive (adversarial order)
- Maximize the sum of weights of vertices in L which got matched.

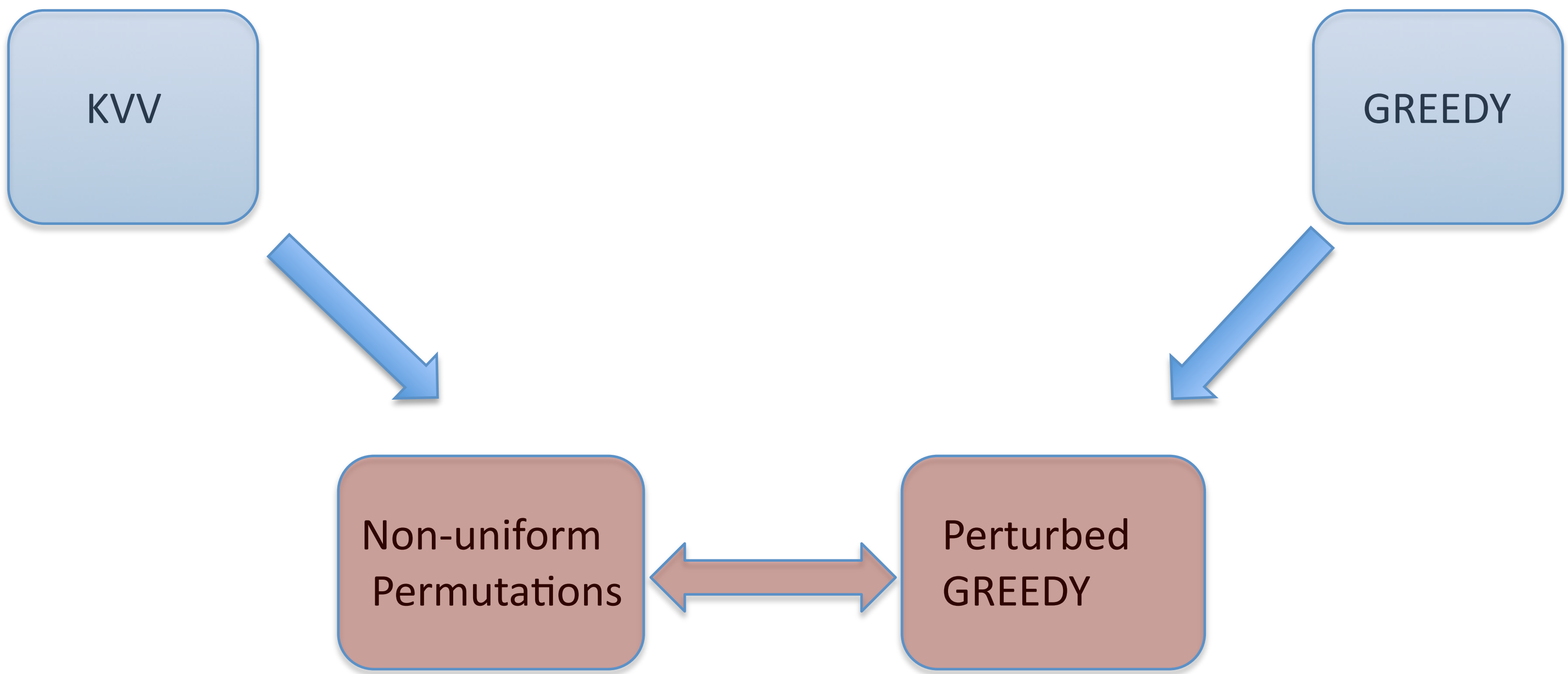
Two Extremes

		
KVV	**3/4 -> 1-1/e**	**1/2 -> 0**
GREEDY	**1/2**	**1**

Intuition



Intuition



A New Algorithm

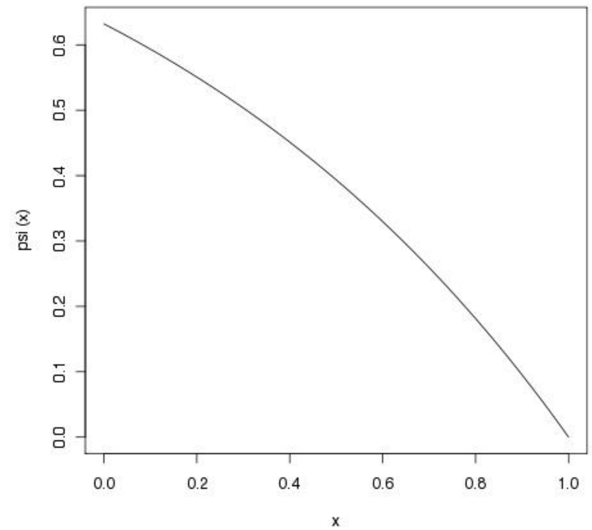
- For each vertex i in L:

  Pick r(i) ← Unif[0, 1] iid

  Define: $W^*(i) = W(i) \times \Psi(r(i))$

- For each arriving vertex in R:

  Pick available neighbor with highest $W^*(i)$



$\Psi(x) := 1 - e^{-(1-x)}$

[Aggarwal, Goel, Karande, Mehta] SODA 2011

A New Algorithm

- For each vertex i in L:

  Pick $r(i) \leftarrow \text{Unif}[0, 1]$ iid

  Define: $W^*(i) = W(i) \times \Psi(r(i))$

- For each arriving vertex in R:

  Pick available neighbor with highest $W^*(i)$

$$\Psi(x) := 1 - e^{-(1-x)}$$

CHECK:

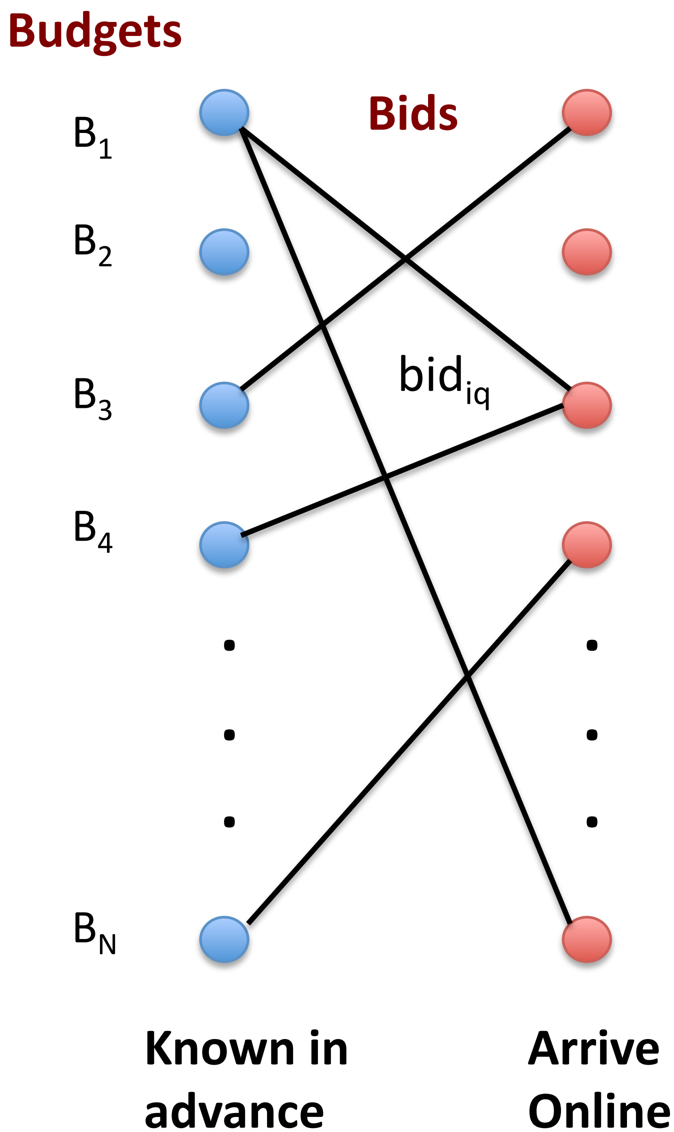
- When all weights equal becomes KVV
- When weights highly skewed becomes GREEDY

Theorem: Factor 1-1/e

[Aggarwal, Goel, Karande, Mehta] SODA 2011

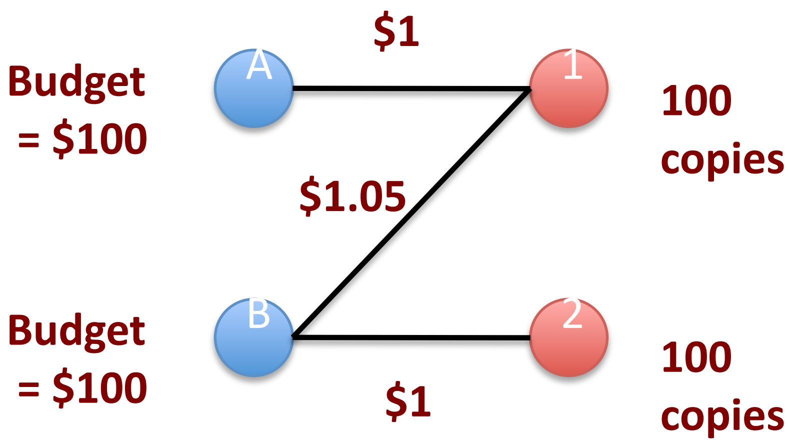
	Adverserial Order	Random order / Unknown iid	Known iid
Bipartite Matching	1-1/e (optimal)	0.655 (0.83)	0.70 (0.83)
Vertex Weighted Matching	1-1/e (optimal)	?	?
Adwords			

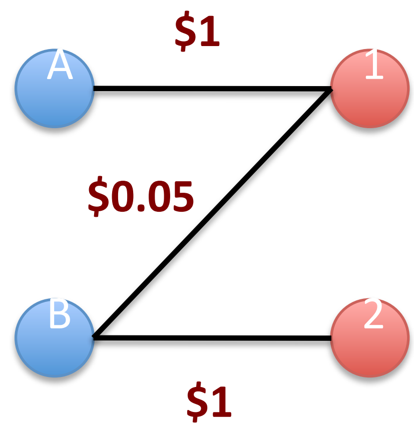
Finally, the "Adwords" problem



- **bid << budget**
- Maximize the sum of budgets spent

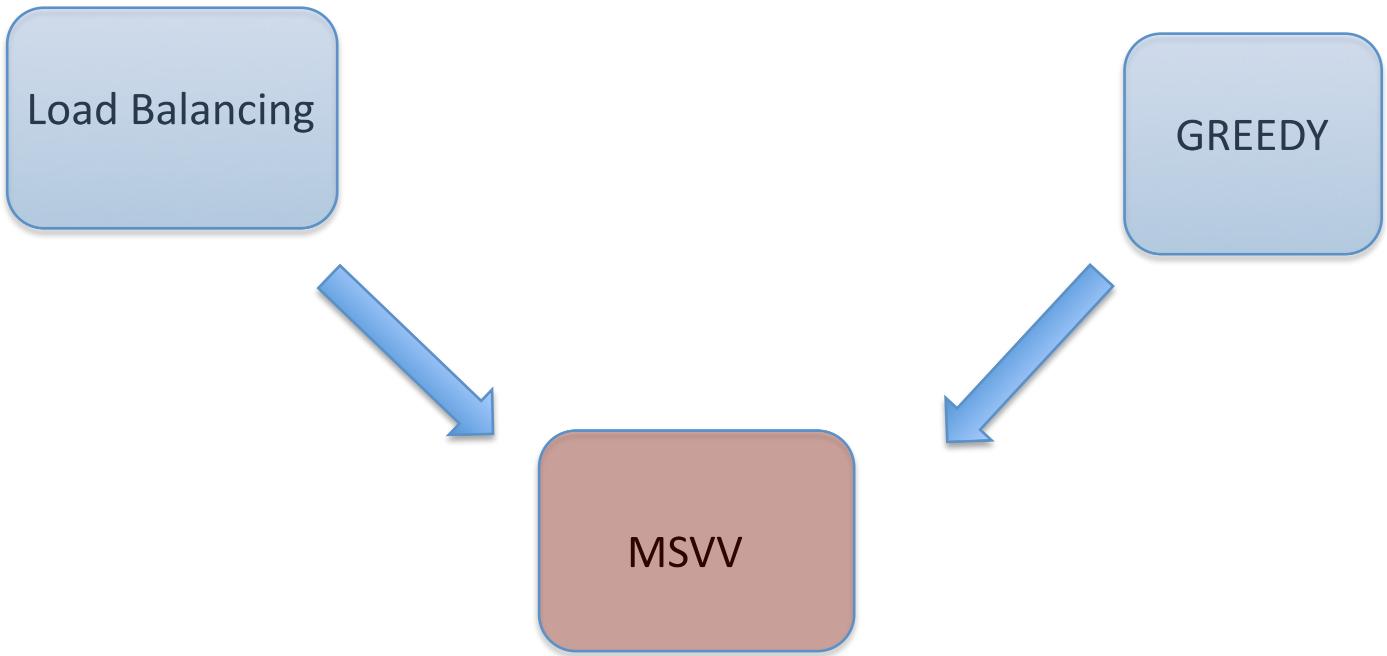
Intuition





Greedy	1/2	1
Load Balance	3/4	1/2

Intuition



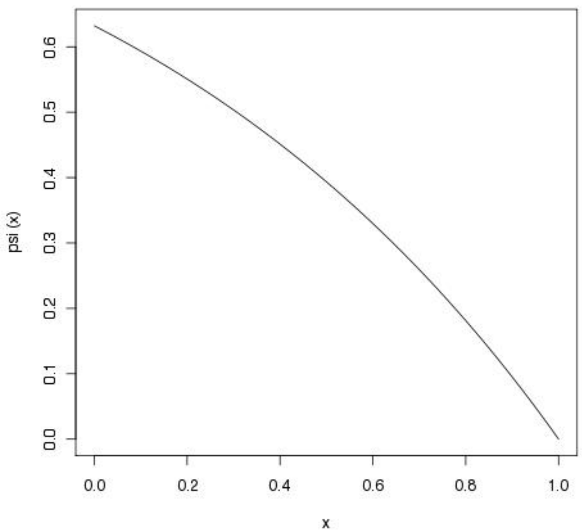
[Mehta, Saberi, Vazirani, Vazirani] FOCS 2005 and J. ACM 2007

Algorithm

- Define:

$bid^*(i, q)$ = bid(i, q) x
Ψ(fraction of budget spent)

- For each arriving vertex in R:
  Pick neighbor with highest $bid^*(i, q)$



Theorem: MSVV achieves 1-1/e. This is optimal.

Algorithm

Recall (for vertex weighted matching):

• Define: $bid^*(i, q) = bid(i, q) \times \Psi(\text{fraction of budget spent})$ • For each arriving vertex in R: Pick neighbor with highest $bid^*(i, q)$	• For each vertex i in L: Pick $r(i) \leftarrow Unif[0, 1]$ iid Define: $W^*(i) = W(i) \times \Psi(r(i))$ • For each arriving vertex in R: Pick available neighbor with highest $W^*(i)$

Theorem: MSVV achieves 1-1/e. This is optimal.

	Adverserial Order	Random order / Unknown iid	Known iid
Bipartite Matching	1-1/e (optimal)	0.655 (0.83)	0.70 (0.83)
Vertex Weighted Matching	1-1/e (optimal)	?	?
Adwords	1-1/e (optimal)		

The tradeoff function

From the optimal dual of the allocation Linear Program (Adwords)

OPT : Uses optimal dual.

Greedy: Uses duals = 0

MSVV: Uses best online duals as a deterministic function of money spent.
Prefix sums of a related LP's dual variables.

[Buchbinder, Jain, Naor] ESA 2007 ***Online Primal Dual Method***

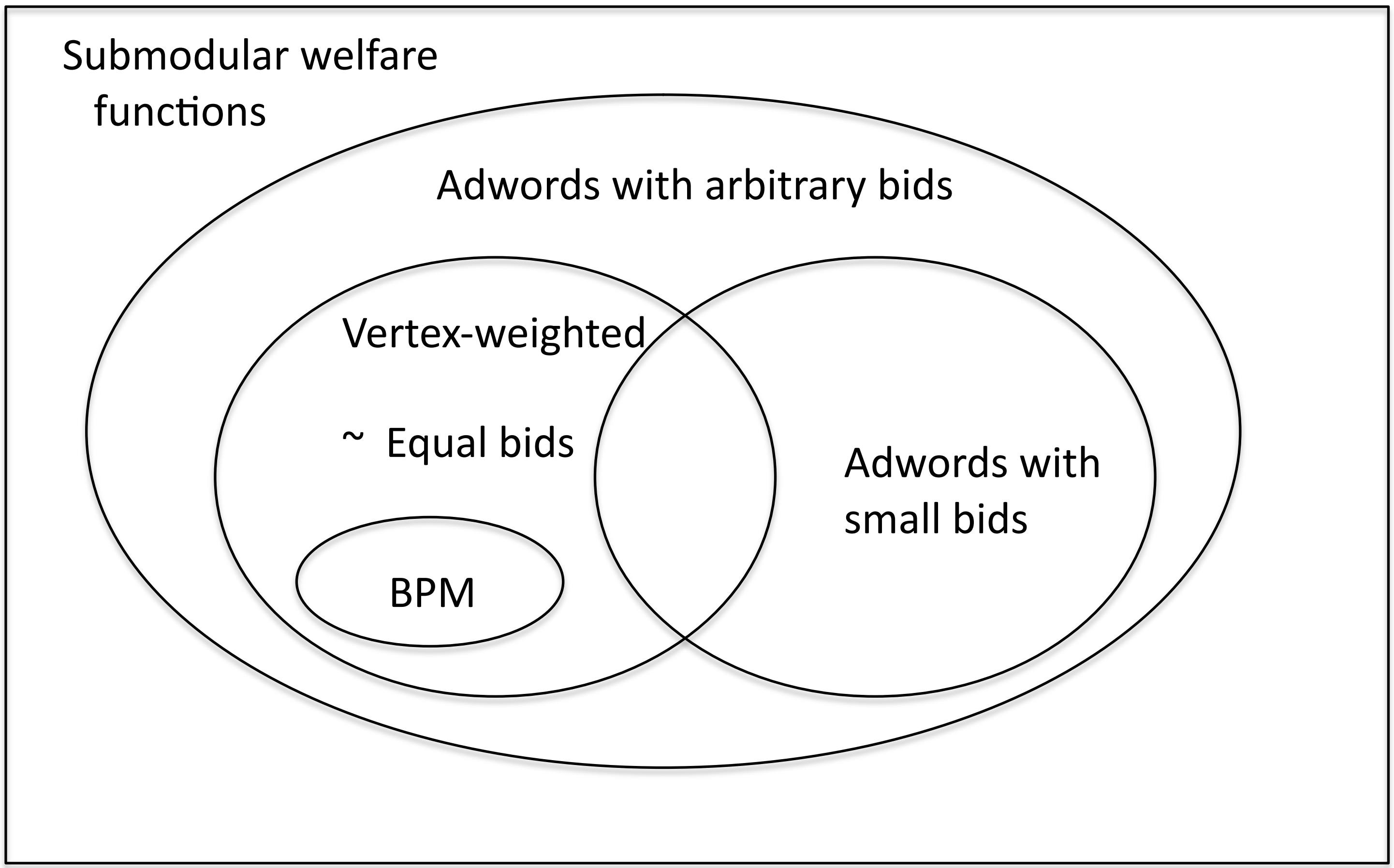
[Devanur Hayes] EC 2009 In random order input we can approximate optimal duals
=> 1 – epsilon

[FHKMS] ESA 2010 Extend to packing problems, describe experiments on real datasets.
[Agrawal, Wang, Ye]

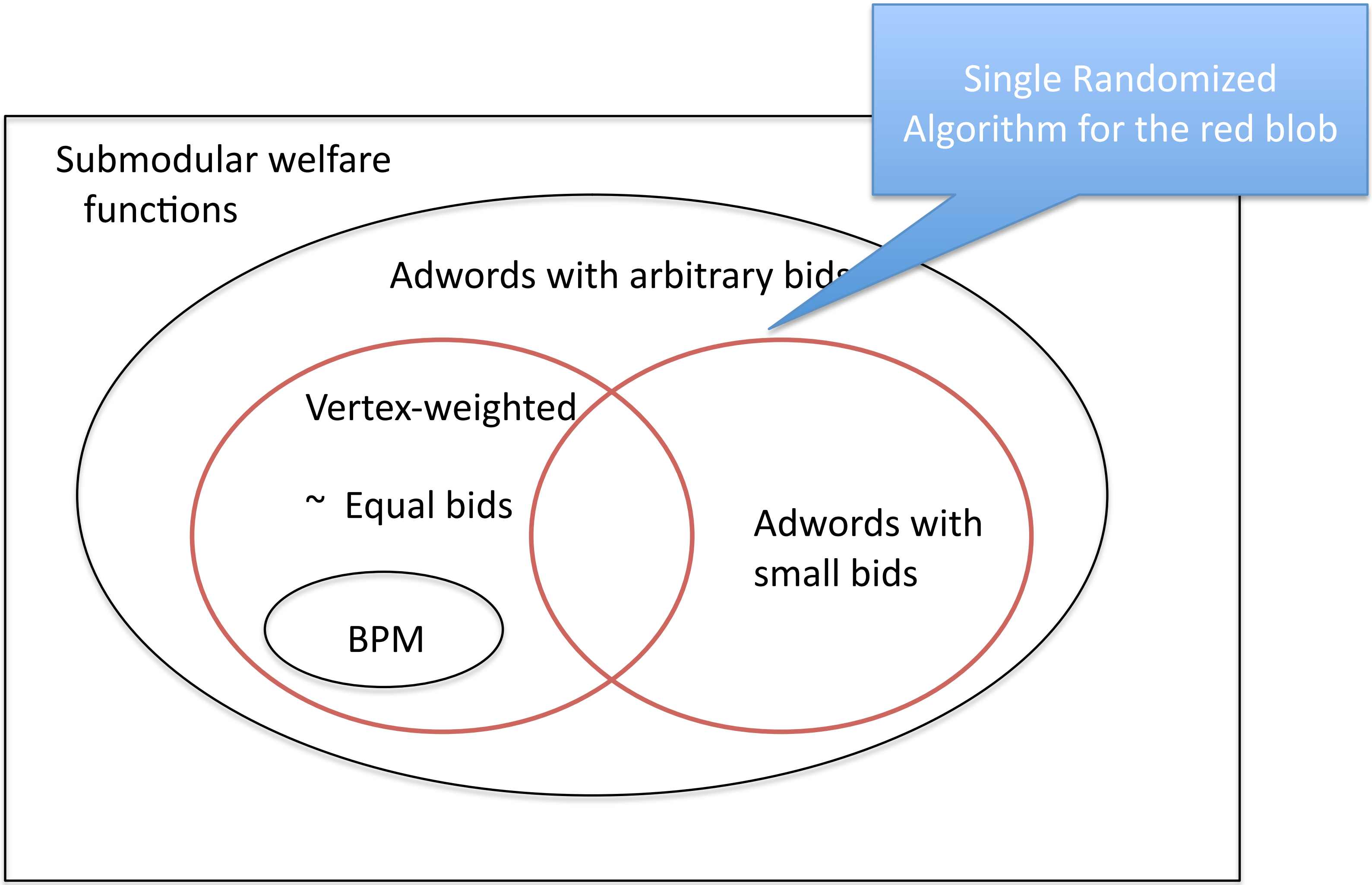
	Adverserial Order	Random order / Unknown iid	Known iid
Bipartite Matching	1-1/e (optimal)	0.655 (0.83)	0.70 (0.83)
Vertex Weighted Matching	1-1/e (optimal)	?	?
Adwords	1-1/e (optimal)	1- epsilon	1 - epsilon

Greedy = 1-1/e
[Goel-Mehta ‘08]

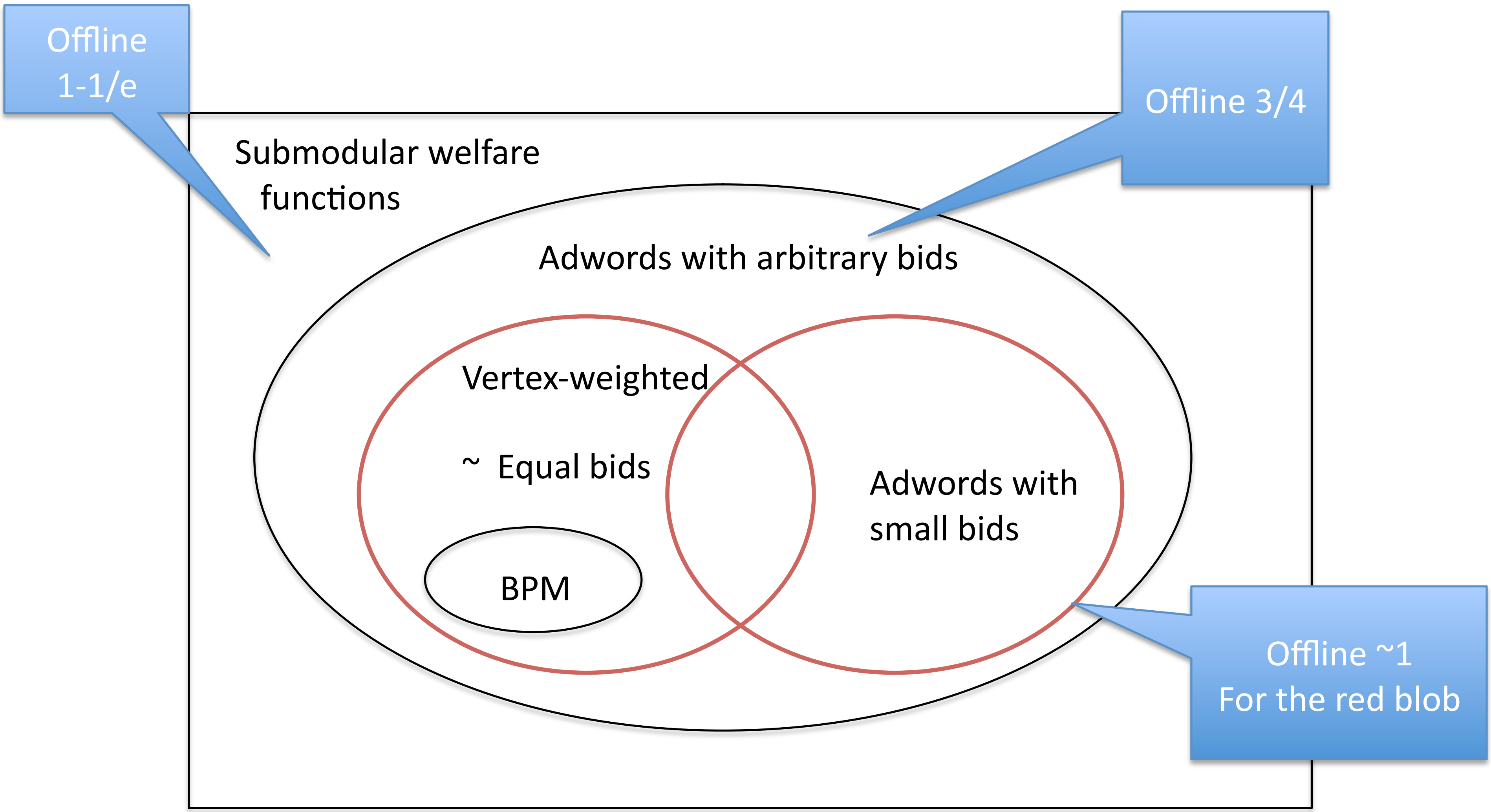
Landscape of problems:



Landscape of problems:



Landscape of problems:



Open Questions

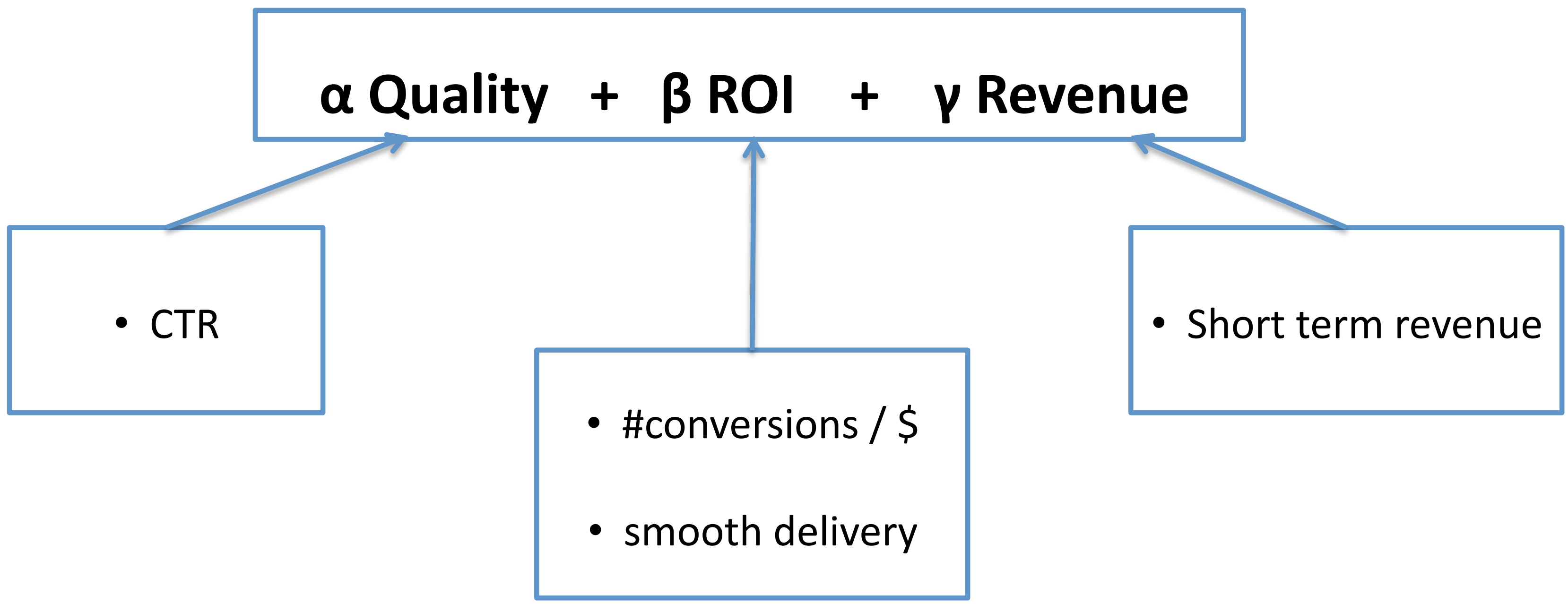
- All the “?” in the table + close the bounds

- Extend these algorithms to work for large bids, submodular?
    - At least beat 1/2?

- Is there a connection to Multiplicative Update Algorithms?
    - Greedy = exploitation
    - Randomize / Deterministic Scaling = regularization
    - [BJN] uses multiplicative updates

Theory to Practice

Objective Functions



Choose α, β, γ : One good business policy "Users come first".

Efficiency in Bid * CTR is a good proxy for all these.

Applying the algorithms

Use best available information

No Info

Full Info

Applying the algorithms

Use best available information

No Info

Distributional input:

Estimate the dual variable from yesterday's logs
Use them for today's allocation
Estimate distribution of metrics rather than items.

What if distributions changes?

Heuristic: increase weights if behind schedule, decrease if ahead of schedule

Full Info

[Devanur, Hayes EC09]
[Mahdian, Nazerzadeh, Saberi EC 07]
[Feldman et al. ESA 2010]
[Kothari, Mehta, Srikant. Manu.]

THANKS