Efficient Online Ad Serving in a Display Advertising Exchange

Kevin Lang (langk@yahoo-inc.com), Joaquin Delgado (jdelgado@yahoo-inc.com), Dongming Jiang, Bhaskar Ghosh, Shirshanka Das, Amita Gajewar, Swaroop Jagadish, Arathi Seshan, Chavdar Botev, Michael Binderberger-Ortega, Sunil Nagaraj, Raymie Stata

Outline

1. Introduction to the RMX Exchange, and a Novel Variant of the Ad-Selection Problem
2. A Graph-based Algorithm for this Problem
3. Experiments using Synthetic Graphs
4. Experience with the Production Ad Server

Outline

1. Introduction to the RMX Exchange, and a Novel Variant of the Ad-Selection Problem
2. A Graph-based Algorithm for this Problem
3. Experiments using Synthetic Graphs
4. Experience with the Production Ad Server

Display Advertising

- Display Advertising is a rapidly growing multi-billion dollar business in which advertisers pay for their ads to be displayed on publishers' web pages.
- It is an efficient way for advertisers to spend their money since they can
  - define targeting logic so that a given ad is only eligible to be shown when specific kinds of users visit specific kinds of web pages,
  - define bidding rules so that the bid for showing an ad depends on properties of the current user and web page.

Display Advertising (cont'd)

- As mentioned in Wednesday's tutorial, display advertising is currently sub-divided into
  - a "futures" mode in which expected collections of opportunities are defined and sold in advance.
  - (this talk) a "spot-market mode" in which a real-time auction amongst advertisers is held at the moment when a user generates a display opportunity by visiting a publisher's web page.

The Players in Display Advertising

- Publishers are business entities who
  - publish web pages
  - sell display opportunities, thus creating “supply”
  - define business logic limiting the possible transactions.
- Advertisers are business entities who
  - have ads that they want to show
  - buy display opportunities, thus creating “demand”
  - define business logic limiting the possible transactions.
- “Ad Networks” are business entities who
  - sign contracts with publishers and advertisers
  - are responsible for running the real-time auctions that bring together the resulting sources of supply and demand thus creating a marketplace.
  - are responsible for enforcing the business logic of all parties so that no undesirable transactions occur.

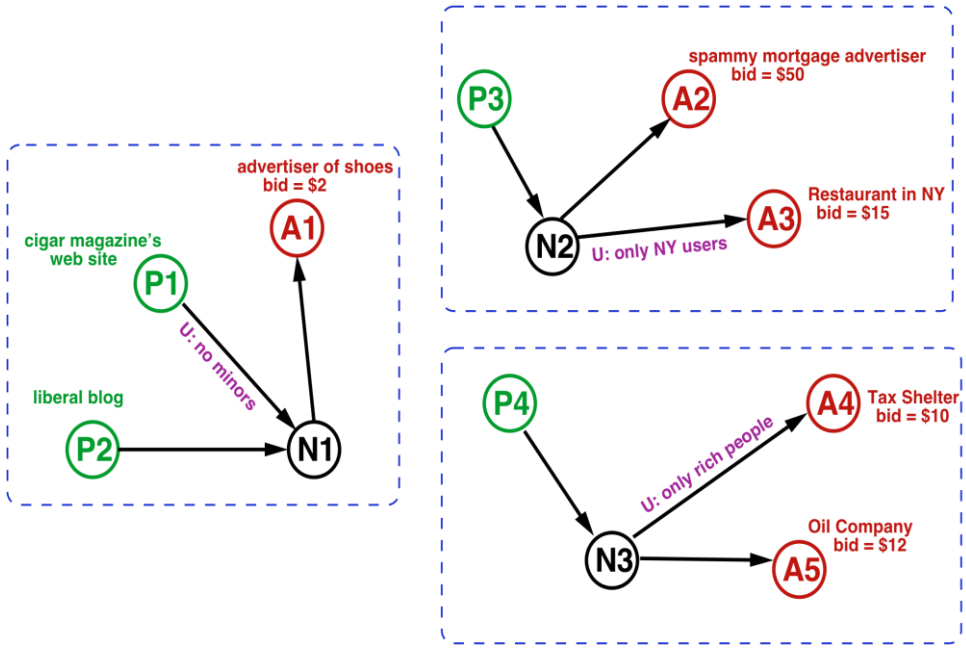
Business Logic Encoded as Predicates

Business logic can be encoded by predicates whose inputs are the properties of objects belonging to various domains, and whose output is true if a transaction involving the tested object would be permissible.

- U-domain predicates test properties of the user.
  - require(user with high income)
  - forbid(user is too young)
- S-domain predicates test properties of the supply.
  - require(topic of web page is sports)
  - forbid(web page is from business competitor)
- D-domain predicates test properties of the demand.
  - forbid(format of ad is flash).
  - forbid(topic of ad is associated with spammers)

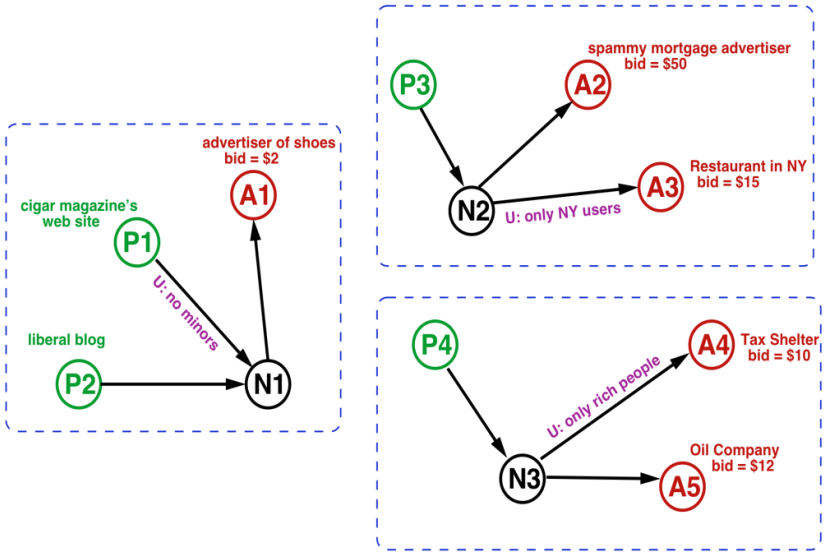
A Graph-based Model of the Story So Far

- Nodes represent publishers, advertisers, and ad networks.
- Edges represent business agreements, specifically which pubs and advertisers have enlisted the real-time matching services of which ad networks.
- The business logic which protects the agreements is modeled by U-, S-, and D-domain predicates on the graphs' edges.



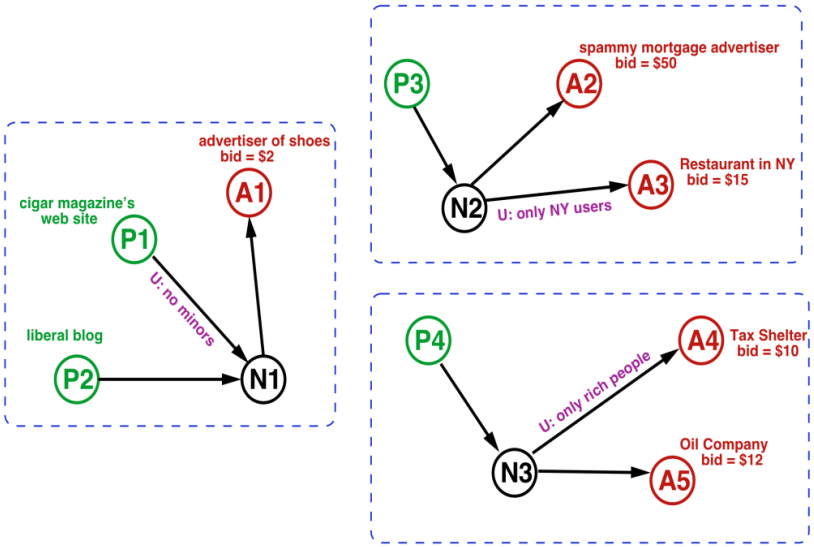
Problems with the World as Described So Far

1. Most Ad Networks would rather not deal with the complexity and expense of building a real-time ad matching service from scratch.
2. Publishers and Advertisers are negatively impacted by the marketplace fragmentation and resulting low liquidity caused by the existence of hundreds of separate Ad Networks.
   - Often the most economically advantageous match cannot be made because a publisher and advertiser belong to different ad networks.



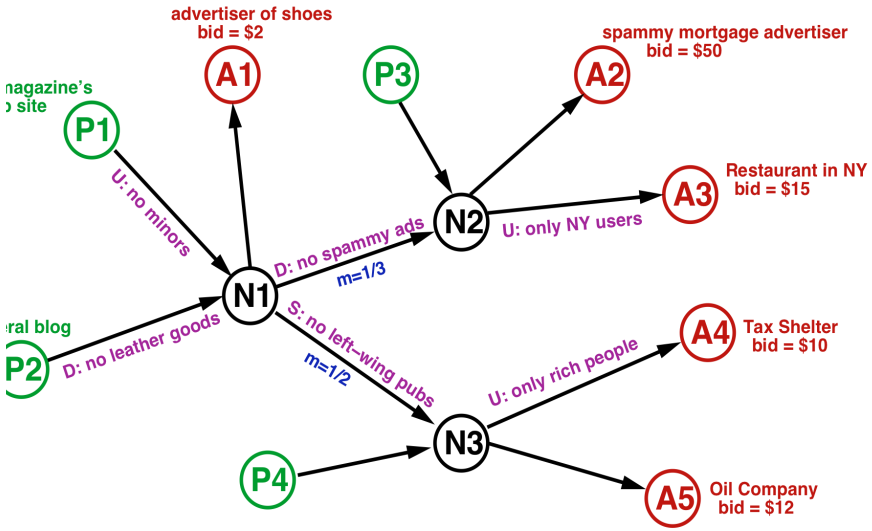
Yahoo's RMX Exchange (1)

- Right Media was a startup company that successfully addressed both of those problems by building the RMX Exchange.
- The first problem is addressed by constructing an ad-serving ASP on which Ad Networks can conveniently run their businesses.
- The problem of marketplace fragmentation is addressed by *opportunity-forwarding agreements* between the ad networks that are using the Exchange.

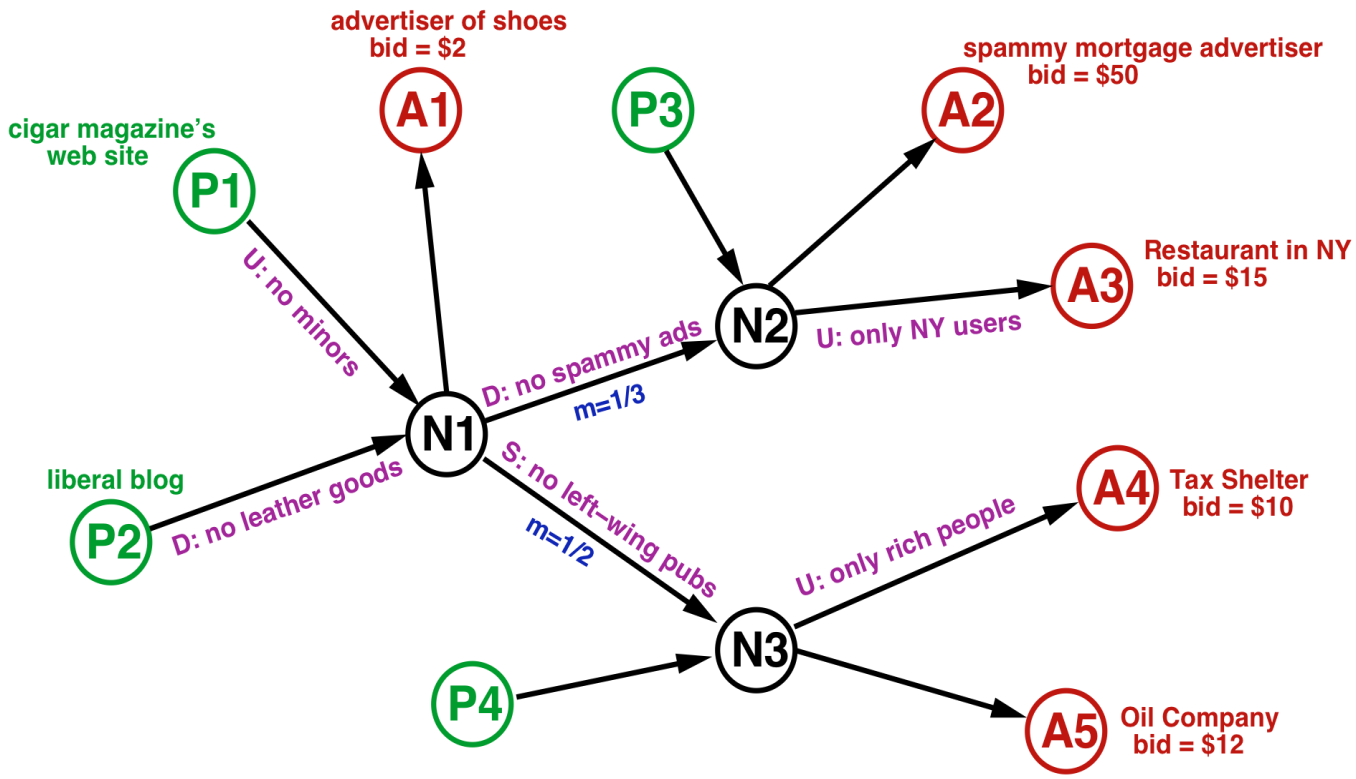


Yahoo's RMX Exchange (2)

- These opportunity forwarding agreement will be modeled by edges between Ad Networks.
- Business logic will be modeled by U-, S-, and D-domain predicates on the edges.
- The agreements include revenue-sharing rules that will be modeled by multipliers on the edges.
- Auction winner is now determined by money reaching the publisher.
- The server's task is to compute the **optimal valid path** to an Ad.



A Toy Example



- Middle-income 40-yr old New York user visits web site P1.
- Valid candidates are P1-N1-A1, P1-N1-N2-A3,P1-N1-N3-A5.
- Winning candidate is P1-N1-N3-A5, causing P1 to get $6.

Formal Problem

- Directed **Multi-Graph** *G*
- Nodes represent ad networks publishers, and advertisers.
- Pubs and advertisers are respectively sources s_i and sinks d_j.
- Arcs represent opportunity-forwarding agreements.
- Arcs are protected by predicates over three domains S,U,and D.
- Each query is a (user,publisher) pair (u_q, s_q)
- **The answer is the optimal valid path *p* from src s_q to any sink d_j.**

- $L_U(u_q, e)$ is a User-domain predicate encoding whether edge e is valid for user u_q.
- $L_S(s_i, e)$ is a Supply-domain predicate encoding whether edge e is valid in paths from source s_i.
- $L_D(d_j, e)$ is a Demand-domain predicate encoding whether edge e is valid in paths to sink d_j.
- $\text{Valid}(u_q, p) = \bigwedge_{e \in p}(L_U(u_q, e) \wedge L_S(s_q, e) \wedge L_D(d_j, e)$
- $B(u_q, d_j)$ is the bid for selecting sink d_j (which represents an ad) for a query representing an opportunity to show an ad to user u_q.
- $m(e)$ is the "revenue sharing" multiplier for edge e.
- $\text{Score}(u_q, p) = B(u_q, d_j) \prod_{e \in p} m(e)$ is the objective function; it is the money that would be received by the publisher after the ad network intermediaries have taken their cuts.
- **ExchangeAdServe**: Given a graph G and a query (s_q, u_q), find a simple path p in G from the query's source s_q to some sink $d_j \in \mathcal{D}$, such that $\text{Score}(u_q, p)$ is maximized subject to $\text{Valid}(u_q, p)$ being true.

Outline

1. Introduction to the RMX Exchange, and a Novel Variant of the Ad-Selection Problem
2. A Graph-based Algorithm for this Problem
3. Experiments using Synthetic Graphs
4. Experience with the Production Ad Server

Plausible-Sounding but Impractical Ideas:

Impractical idea inspired by the fact that *in principle* one could use a standard ad matching algorithm with paths playing the role of ads.

- **Pre-generate** all possible paths and index them according to the boolean expressions on the paths.
- At ad-call time**, fetch the set of valid paths** for that opportunity, then get bids and select the winner.
- Flaw: requires *too much space, because too many paths*.

Semi-practical idea.

- At ad-call time, **generate the set of valid paths from scratch**, then get bids and select the winner.
- Flaw: requires *too much time, because too many paths*.
- This is “Algorithm A” used when the Exchange was a startup.

Some guiding principles for a better solution

A. For a scalable solution, never ever generate an explicit representation of a complete set of paths, valid or otherwise.
B. Instead, take advantage of the fact that a graph is a compact representation of the complete set of paths that are in it.
C. Following database intuition, try to rapidly winnow the set of candidate paths by enforcing predicates as soon as possible.

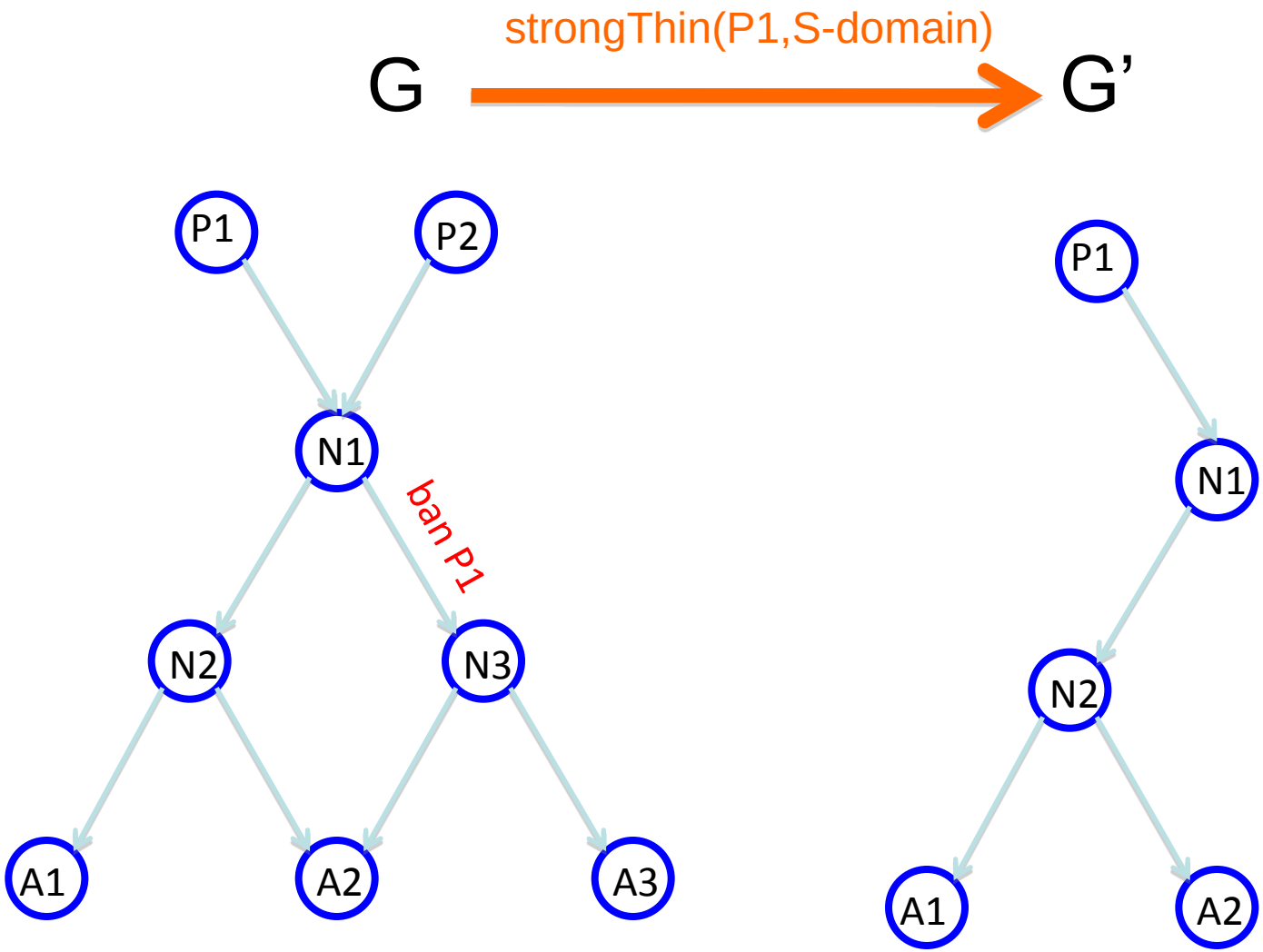
Graph Thinning

A. For a scalable solution, never ever generate an explicit representation of a complete set of paths, valid or otherwise.
B. Instead, take advantage of the fact that a graph is a compact representation of the complete set of paths that are in it.
C. Following database intuition, try to rapidly winnow the set of candidate paths by enforcing predicates as soon as possible.

Principles B and C above lead to the concept of **graph thinning.**
For function strongThin(srcNode,domain), which is applicable to domains U and S but not domain D, the steps are:

1. Evaluate predicates of the given domain.
2. Delete invalid edges.
3. Perform connectivity analysis from srcNode using BFS.
4. Delete unreachable edges and nodes.

Example of Strong Thinning for Supply Domain



Sketch of Algorithm B

1. G’ <- strongThin (G, s, (U*S)-domains)
2. Get bids for each ad d_j in G’
3. Sort ads in decreasing order of bid
4. For each d_j in sorted order:
   - If legalPathExists (G’, s, d_j, D-domain) then break.

Note: we are temporarily pretending that ad networks do not take shares of the money, so advertiser bids determine the auction winner.

Note: legalPathExists can be a predicate-checking BFS.

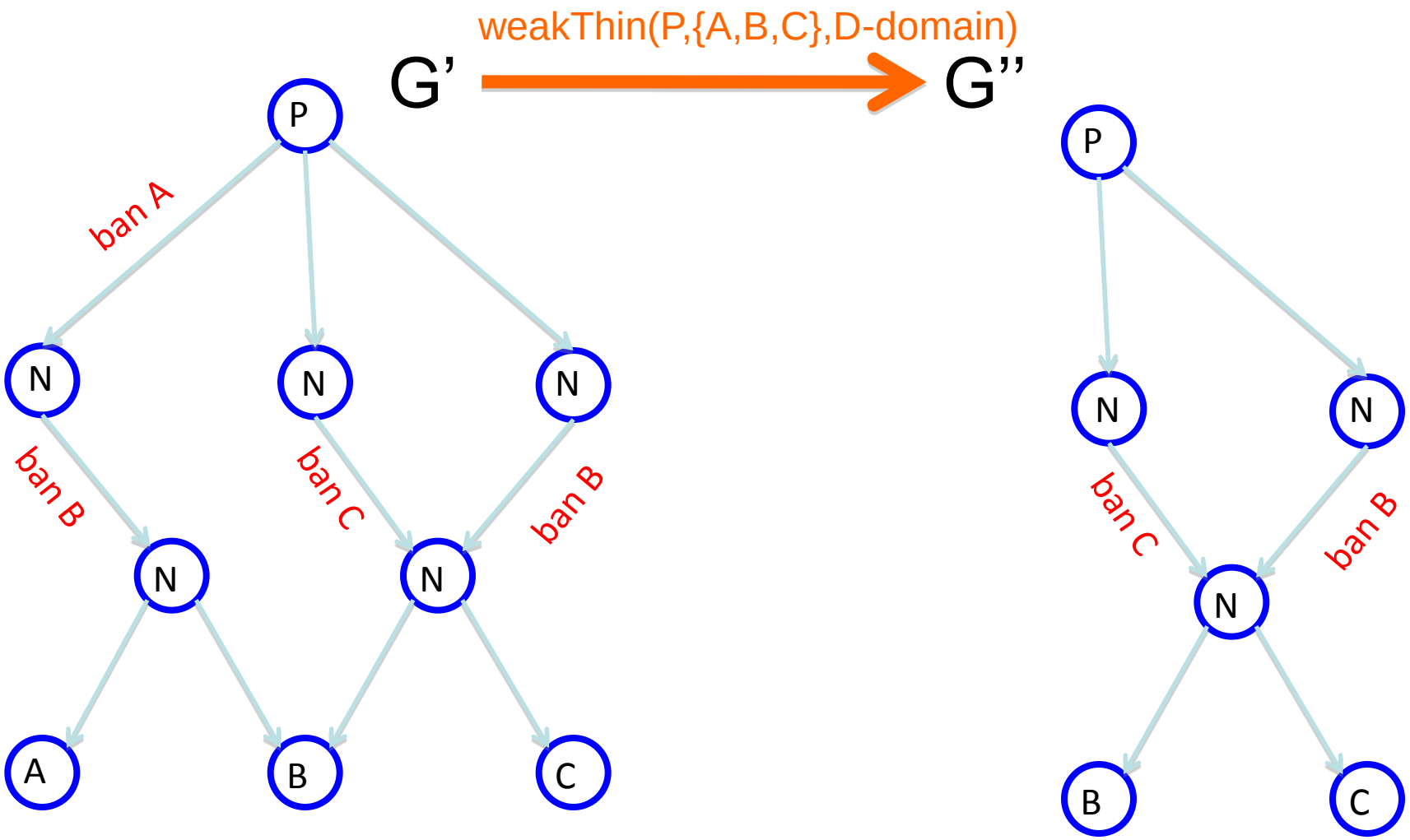
Question about Algorithm B

1. G’ <- strongThin (G, s, (U*S)-domains)
2. Get bids for each ad d_j in G’
3. Sort bids in decreasing order of bid.
4. For each d_j in sorted order:
   - If legalPathExists (G’, s, d_j, D-domain) then break.

Can we do the graph thinning even earlier, say in a pre-computation?

- Yes, for the S- and D-domains, where the object attributes are known before any ad call occurs.
- *However*, for the D-domain the result cannot be represented by a predicate-free subgraph of G, so we use **weakThin**.

Example of Weak Thinning for Demand Domain



Sketch of Algorithm B with Pre-computation

Offline for source s.

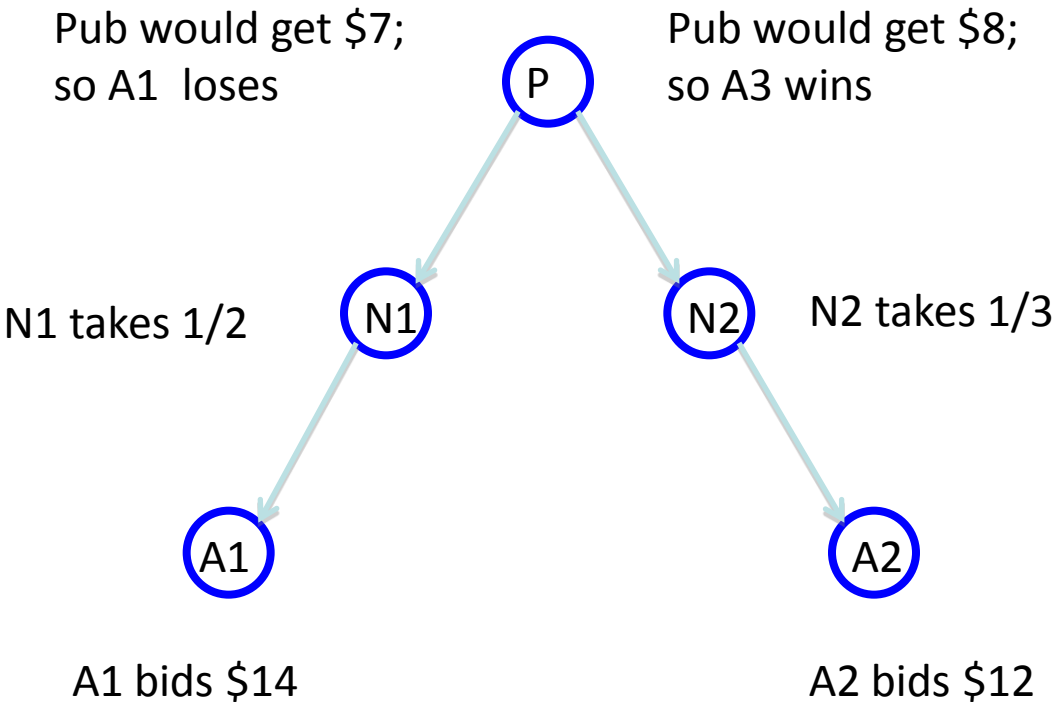
A. G'(s) <- strongThin (G, s, S-domain)
B. G''(s) <- weakThin (G'(s), s, $\{d_j\}$, D-domain)

Online for query (s,u)

1. G''' <- strongThin (G''(s), s, U-domain)
2. Get bids for each ad d_j in G'''
3. Sort ads in decreasing order of bid
4. For each candidate ad:
   - If legalPathExists (G''', s, d_j, D-domain) then break.

Revenue sharing can change the auction winner.

Actually, the ad networks each take a share of the money, and it is the money reaching the publisher that decides the auction winner, not the raw bids from the advertisers.



Two Changes to Support Revenue Sharing

•Use Dijkstra instead of BFS for path finding so that if the thinned graph contains *any* valid path then we get the *best* one, i.e. the path that transmits the largest fraction of the bid to the publisher.

•The loop over candidate ads cannot stop at the *first* ad with a valid path, but must instead wait for a candidate ad whose best valid path results in a publisher payment that exceeds upper bounds on the publisher payments of all remaining candidates.

- The UB's are derived from *provisional best paths* found by Dijkstra in a thinned graph that respects some but not all constraints.
- These are UB's because *final best paths* that respect all constraints can be worse but not better.

Algorithm B with Pre-comp and Revenue Sharing

Offline for source s.

A. G'(s) <- strongThin (G, s, S-domain)
B. G''(s) <- weakThin (G'(s), {d_j}, D-domain)

Online for query (s,u)

1. G''' <- strongThin (G''(s), s, U-domain)
2. Get bids for reachable ads. Run one-to-many Dijkstra from s to get path UB's. Multiply by bids to get UB's on payments to pub.
3. Sort candidate ads in decreasing order of UB's.
4. LB <- 0
5. For candidate ads in sorted order
   - If LB > UB's of all remaining candidates then break.
   - If DijkstraLegalPathExists (G''', s, d_j, D-domain) then
     - if (ad,path) provides new best payment then raise LB.

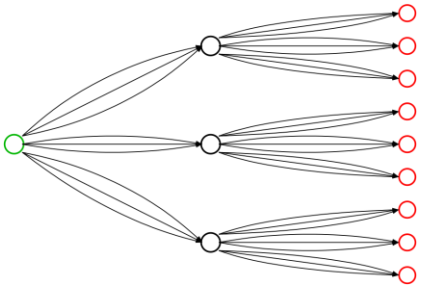
Computational Complexity

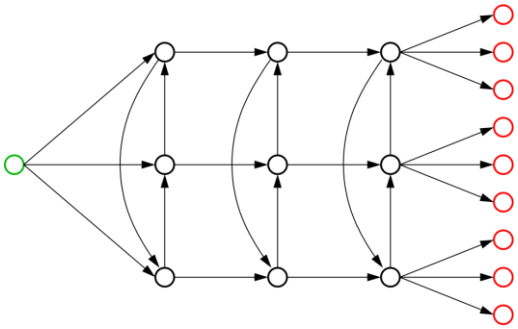
- Worst case running time of Algorithm B is trivially seen to be $O(|\{d_j\}| * |E| * \log |N|)$,
  - In contrast, Algorithm A can require exponential time.
- The worst case would occur if the graph thinning and candidate ordering heuristics were completely defeated.
  - We emphasize that the heuristic nature of this algorithm relates to its running time, not the optimality of its output.
- If the worst case occurred in practice, the algorithm would be useless for meeting latency targets.
- Therefore a more interesting question is “typical case” running time, which can be investigated experimentally.

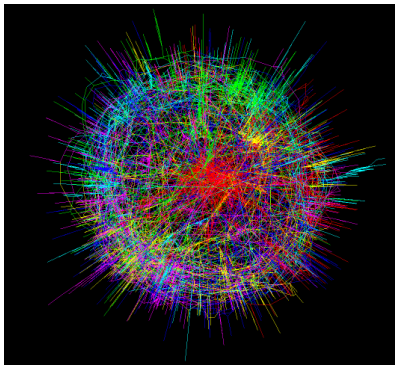
Outline

1. Introduction to the RMX Exchange, and a Novel Variant of the Ad-Selection Problem
2. A Graph-based Algorithm for this Problem
3. Experiments using Synthetic Graphs
4. Experience with the Production Ad Server

Experiments on Synthetic Graphs







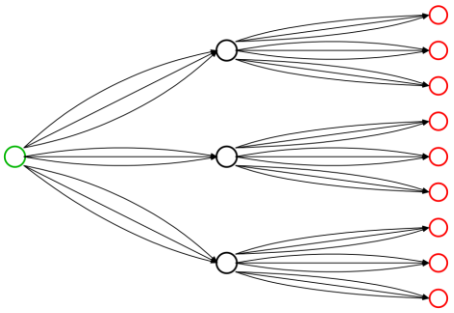
We studied the scaling properties of the old (A) and new (B) path selection algorithms using 3 parameterized families of synthetic graphs with diverse connectivity properties.

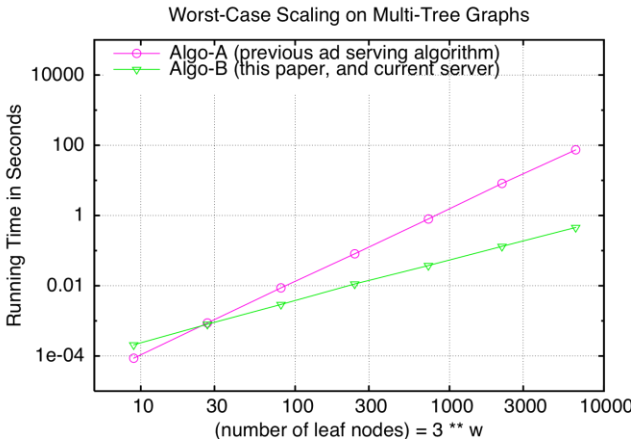
- trees with parallel edges
- grid-like directed graphs containing cycles
- expander-like random graphs

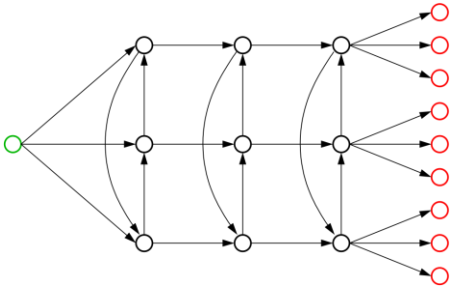
In particular, we varied

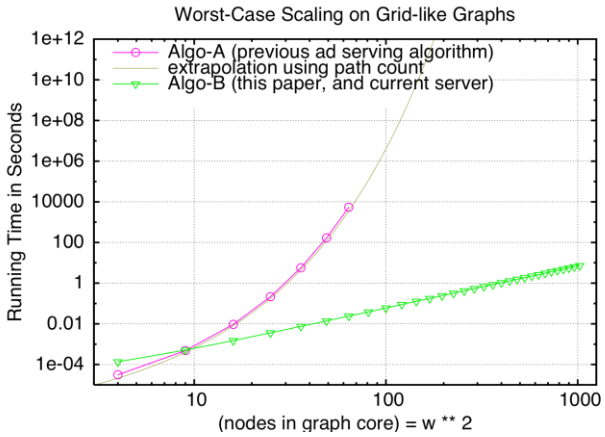
- the size of the graphs
- the selectivity of the predicates of each domain.

Scaling with Predicates that Induce Worst Case





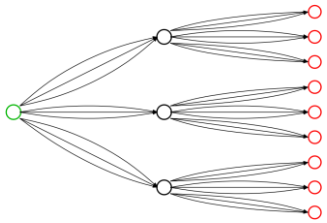


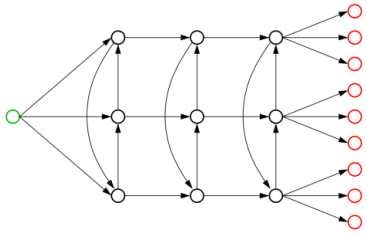


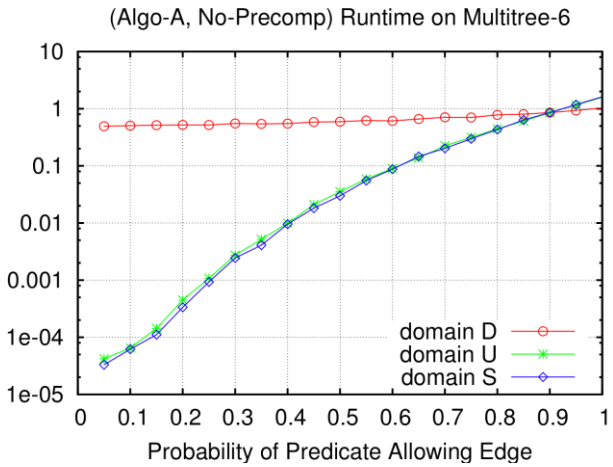
Not too surprisingly,
the polynomial-time algorithm B scales better than the exponential-time algorithm A.

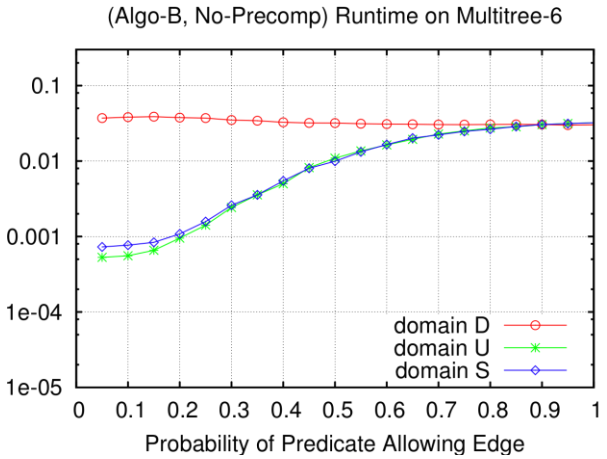
Variable-Selectivity Random Predicates

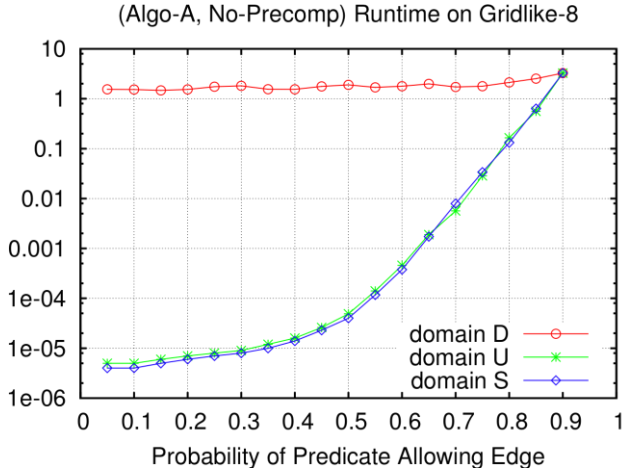
These non-worst case experiments might be a better of probe of relevant algorithm behavior.

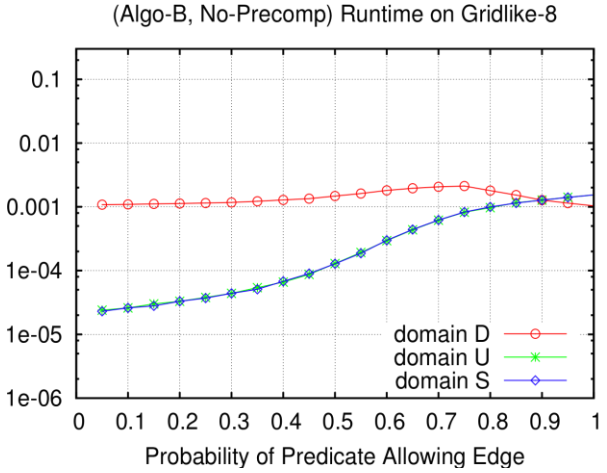








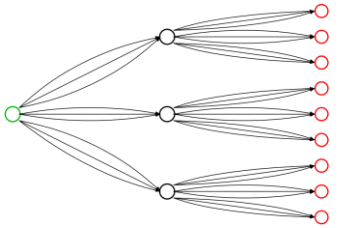


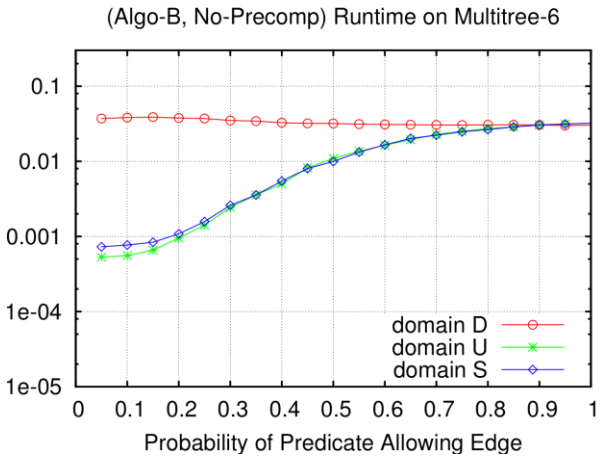


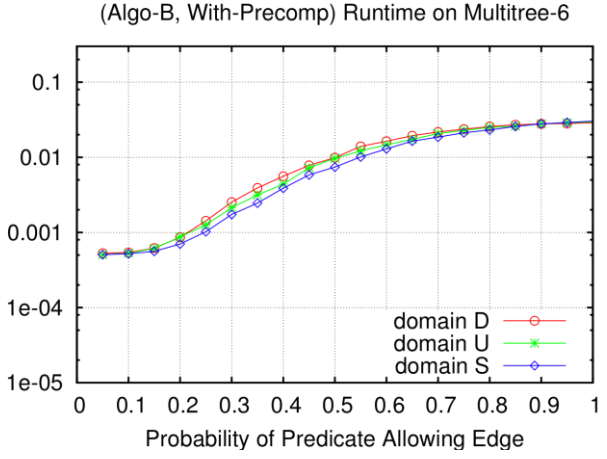
Observations:

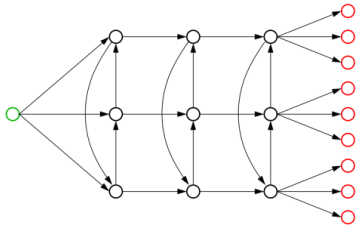
- algorithm B shows a smaller range of latency variation
- there is no cost reduction as the D-domain becomes more selective; maybe the pre - computation would help.

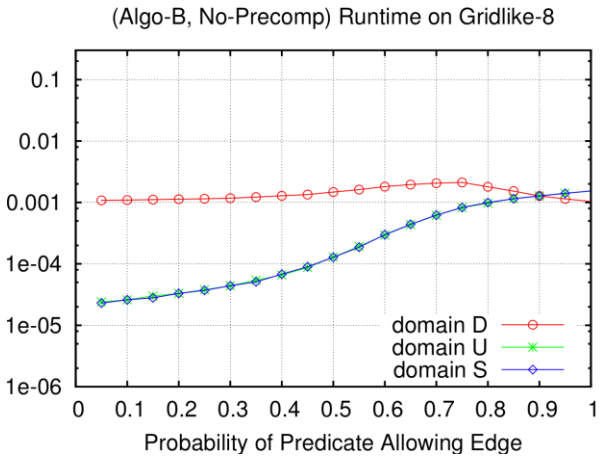
Variable-Selectivity Random Predicates (cont'd)

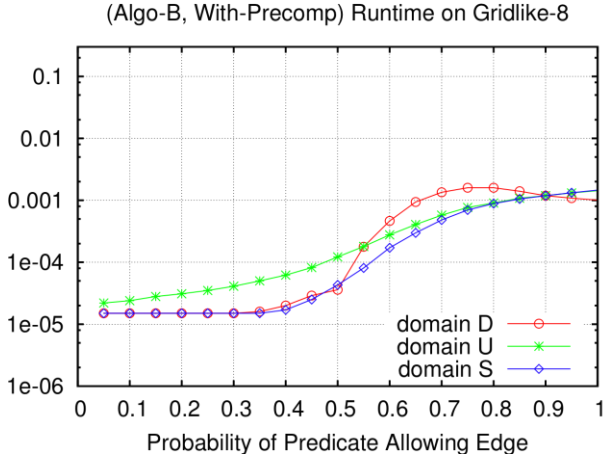








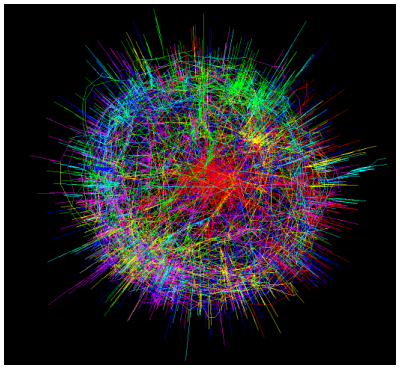


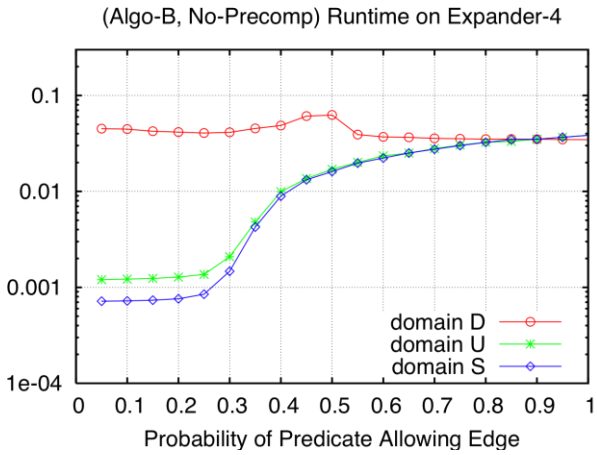


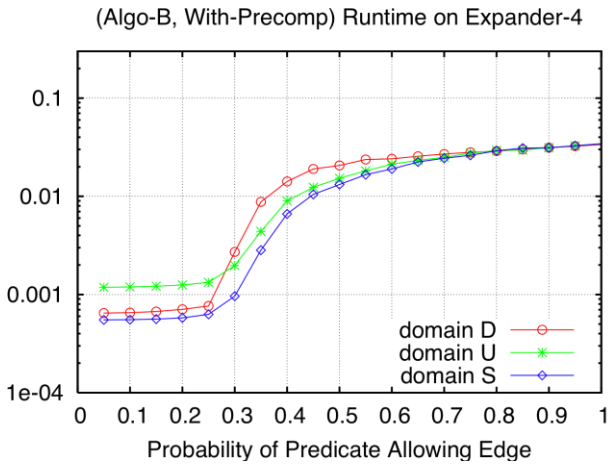
With the pre-computation, Algo-B does get faster as the D-domain predicates become more selective, even though we are only using weakThin() for the D-domain.

Results for Random Expander-like Graphs

The exponential-time Algorithm-A is hopeless in this case, but for Algorithm-B we get results that are similar to the earlier ones, thus illustrating the robustness of this algorithm.







Outline

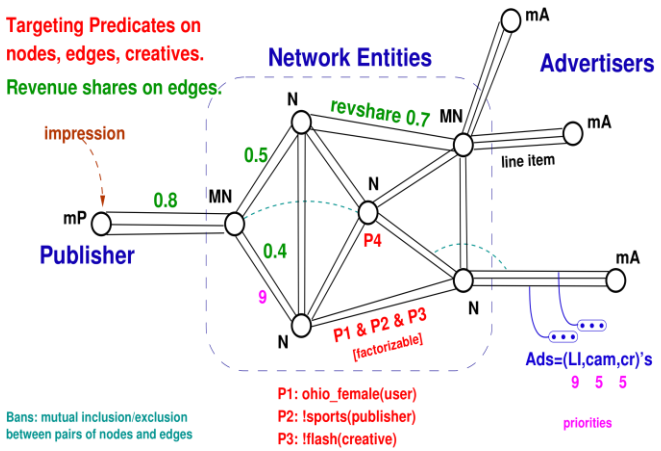
1. Introduction to the RMX Exchange, and a Novel Variant of the Ad-Selection Problem
2. A Graph-based Algorithm for this Problem
3. Experiments using Synthetic Graphs
4. Experience with the Production Ad Server

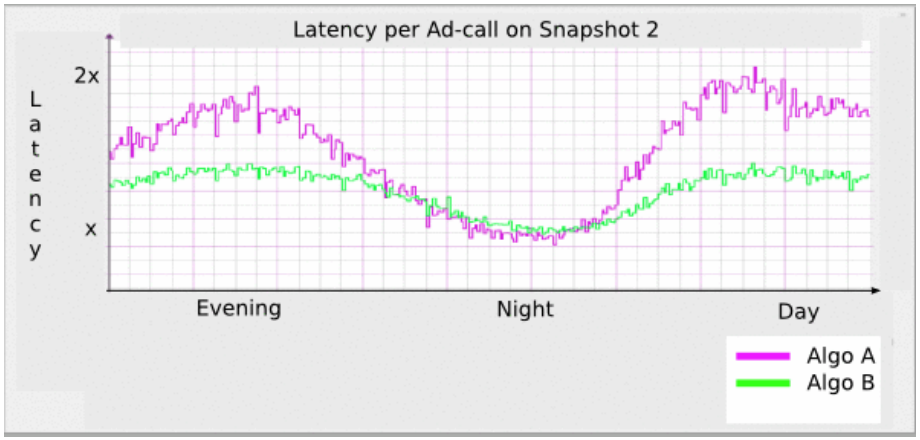
The Actual Yahoo! Exchange

- The Yahoo! Exchange is the largest of its kind, with hundred of thousands of business entities represented in the graph.
- The Exchange handles billions of transactions per day, with latencies on the order of milliseconds.
- The production ad servers are now based on this paper's algorithm B.

Actual Ad Serving

•The behavior of the real ad servers is consistent with the results given above for synthetic graphs, despite the fact that the full task is complicated by the presence of numerous additional exchange features.





•For example, the new ad server based on algorithm B exhibits a smaller range of diurnal latency variation than the old server that was based on algorithm A.

Current and Future Work

1. Further speedup by indexing edge predicates.
   - A Yahoo! paper on this topic will be at WWW2011.
2. RTB (Real Time Bidding) is an industry trend in which opportunity-forwarding occurs between exchanges.
   - The effective marketplace size and liquidity grows.
   - Latency SLA's decrease since multiple Exchange's are communicating over the internet.
   - The complete-knowledge assumption underlying simple pre-computation schemes is violated.

End of Talk

FIN