Online Matching and Adwords

Aranyak Mehta

Google Research
Mountain View, CA
Search Advertising (Adwords)
users → Queries (online) → Search engine → Bidding (offline) → advertisers
Search engine

Find matching candidates

Score candidates

Run auction

Queries (online)

Bidding (offline)
• Advertiser Budgets => Demand Constraint => Matching
Display ads

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

- **Supply is online** from page views
Ad Exchanges

Publisher ➔ Ad Exchange ➔ Publisher
Publisher ➔ Ad Exchange ➔ Publisher
Publisher ➔ Ad Exchange ➔ Publisher

Ad Network ➔ Ad Exchange ➔ Ad Network
Ad Network ➔ Ad Exchange ➔ Ad Network
Ad Network ➔ Ad Exchange ➔ Ad Network

... 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

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

Known in advance

Arrive Online
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

\[ u \text{ miss @ } t \]

\[ \Rightarrow u^* \text{ match @ } s < t \]

\[ \Rightarrow \text{Factor } \geq 1/2 \]
Analysis of KVV in 1 slide

[\textit{Birnbaum, Matheiu}] SIGACT News 2008

u miss @ t

\Rightarrow \text{In each of the n permutations}
\quad u^* \text{ match at } s \leq t
Analysis of KVV in 1 slide

[Birnbaum, Matheiu] SIGACT News 2008

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

\Rightarrow \text{Factor 1-1/e}
Tight example
Goal: “Adwords” problem

- bid $\ll$ budget
- Maximize the sum of budgets spent

Budgets

- $B_1$
- $B_2$
- $B_3$
- $B_4$
- $B_N$

Known in advance

Arrive Online
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.
<table>
<thead>
<tr>
<th>Adversarial Order</th>
<th>Random order / Unknown iid</th>
<th>Known iid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bipartite Matching</td>
<td>1-1/e (optimal)</td>
<td></td>
</tr>
<tr>
<td>Vertex Weighted Matching</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adwords</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Unknown Distribution / Random Order

- GREEDY = 1-1/e
- KVV ≥ 1-1/e
- Can we do better? How about KVV itself?
• 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
  \[ 1 - \frac{1}{\sqrt{k}} \]

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

R

0.75

0

L

R

0.726

0

L
<table>
<thead>
<tr>
<th></th>
<th>Adverserial Order</th>
<th>Random order / Unknown iid</th>
<th>Known iid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bipartite Matching</td>
<td>1-1/e (optimal)</td>
<td>0.696 (0.83)</td>
<td></td>
</tr>
<tr>
<td>Vertex Weighted Matching</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adwords</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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?
**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.

⇒ **Factor 1-1/e**
Attempt 2: Power of two choices!

**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

*Two Matchings in Base Graph*
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, Oveis-Gharan, Saberi] (SODA 2011): 0.70 and upper bound of 0.83
<table>
<thead>
<tr>
<th></th>
<th>Adversarial Order</th>
<th>Random order / Unknown iid</th>
<th>Known iid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bipartite Matching</td>
<td>1-1/e (optimal)</td>
<td>0.655 (0.83)</td>
<td>0.70 (0.83)</td>
</tr>
<tr>
<td>Vertex Weighted</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matching</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Adwords</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>KVV</th>
<th>GREEDY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>![](3/4 -&gt; 1-1/e)</td>
<td>1/2</td>
</tr>
<tr>
<td>1</td>
<td>1/2</td>
<td>1</td>
</tr>
<tr>
<td>M</td>
<td>1/2 -&gt; 0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table:**
- **KVV:**
  - $3/4 \rightarrow 1 - 1/e$
- **GREEDY:**
  - $1/2$
Intuition

- KVV
  - Non-uniform Permutations

- GREEDY
  - Perturbed GREEDY
Intuition

KVV

Non-uniform Permutations

Perturbed GREEDY

GREEDY
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 $W^*(i)$

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

[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 $1 - \frac{1}{e}$

[Aggarwal, Goel, Karande, Mehta] SODA 2011
<table>
<thead>
<tr>
<th></th>
<th>Adversarial Order</th>
<th>Random order / Unknown iid</th>
<th>Known iid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bipartite Matching</td>
<td>1-1/e (optimal)</td>
<td>0.655 (0.83)</td>
<td>0.70 (0.83)</td>
</tr>
<tr>
<td>Vertex Weighted Matching</td>
<td>1-1/e (optimal)</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Adwords</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Finally, the “Adwords” problem

- bid $\ll$ budget
- Maximize the sum of budgets spent
<table>
<thead>
<tr>
<th>Budget = $100</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$1</td>
</tr>
<tr>
<td>1</td>
<td>$1.05</td>
</tr>
<tr>
<td>2</td>
<td>100 copies</td>
</tr>
<tr>
<td>B</td>
<td>$1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Budget = $100</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>A</td>
<td>$1</td>
</tr>
<tr>
<td>1</td>
<td>100 copies</td>
</tr>
<tr>
<td>B</td>
<td>$1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Greedy</th>
<th>1/2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Balance</td>
<td>3/4</td>
<td>1/2</td>
</tr>
</tbody>
</table>
Intuition

Algorithm

- Define:
  \[ \text{bid}^*(i, q) = \text{bid}(i, q) \times \Psi(\text{fraction of budget spent}) \]

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

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

Recall (for vertex weighted matching):

<table>
<thead>
<tr>
<th>Define:</th>
<th>( \text{bid}^*(i, q) = \text{bid}(i, q) \times \Psi(\text{fraction of budget spent}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>For each arriving vertex in R:</td>
<td>Pick neighbor with highest ( \text{bid}^*(i, q) )</td>
</tr>
</tbody>
</table>

| 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 \( W^*(i) \) |

**Theorem:** MSVV achieves 1-1/e. This is optimal.
<table>
<thead>
<tr>
<th>Problem</th>
<th>Adversarial Order</th>
<th>Random order / Unknown iid</th>
<th>Known iid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bipartite Matching</td>
<td>1-1/e (optimal)</td>
<td>0.655 (0.83)</td>
<td>0.70 (0.83)</td>
</tr>
<tr>
<td>Vertex Weighted Matching</td>
<td>1-1/e (optimal)</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Adwords</td>
<td>1-1/e (optimal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

**[FKMS] ESA 2010** Extend to packing problems, describe experiments on real datasets. **[Agrawal, Wang, Ye]**
<table>
<thead>
<tr>
<th>Problem</th>
<th>Adversarial Order</th>
<th>Random order / Unknown iid</th>
<th>Known iid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bipartite Matching</td>
<td>1-1/e (optimal)</td>
<td>0.655 (0.83)</td>
<td>0.70 (0.83)</td>
</tr>
<tr>
<td>Vertex Weighted Matching</td>
<td>1-1/e (optimal)</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Adwords</td>
<td>1-1/e (optimal)</td>
<td>1- epsilon</td>
<td>1 - epsilon</td>
</tr>
</tbody>
</table>

Greedy = 1-1/e [Goel-Mehta ’08]
Landscape of problems:

- Submodular welfare functions
- Adwords with arbitrary bids
- Vertex-weighted
  - Equal bids
  - BPM
- Adwords with small bids
Landscape of problems:

- Submodular welfare functions
- Adwords with arbitrary bids
  - Vertex-weighted
  - ~ Equal bids
  - BPM
- Adwords with small bids

Single Randomized Algorithm for the red blob
Landscape of problems:

- Submodular welfare functions
- Adwords with arbitrary bids
  - Vertex-weighted
    - ~ Equal bids
    - BPM
  - Adwords with small bids

Offline 1-1/e
Offline 3/4
Offline ~1
For the red blob
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

\[ \alpha \text{ Quality} + \beta \text{ ROI} + \gamma \text{ Revenue} \]

- CTR
- #conversions / $
- smooth delivery
- Short term revenue

Choose \( \alpha, \beta, \gamma \) : 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

**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

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