Design Lab Report

ADWORDS

submitted by
Amatya Sharma (17CS30042)
5th Year Dual Degree
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

Special Thanks to
Prof. Swagato Sanyal
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

November 13, 2021

1 Introduction and Related Works

Google marketplace can be considered as the greatest source of revenue for Google search engine. Users search for specific queries on the engine and the google marketplace appends a ”related” AD as its first result query, followed by the actual results of the query search. The one ”AD” result is what forms the basis of Google Search Engine’s revenue, and in order to maximize that revenue, Google has to optimize the types of ADs shown for specific user search query. This essential gives rise to the problem of ADWORDS (General). Informally, this problem involves matching keyword queries, as they arrive online, to advertisers having budget constraints.

Taking about prior results, for the case with small bids, an optimal algorithm achieving a competitive ratio of $\left(1 - \frac{1}{e}\right)$ was first given in [MSVV07]. Another variant of ADWORDS was recently defined as SINGLE-VALUED which a special case of GENERAL in which each bidder makes bids of a single value for the queries he is interested in. However, the core problem, of which all the problems mentioned above are extensions, is OBM. This problem occupies a central place not only in online algorithms but also in matching-based market design. For OBM, a simple, optimal, randomized online algorithm, called RANKING, was given in [KVV90]. Its competitive ratio is $\left(1 - \frac{1}{e}\right)$ and [KVV90] showed that no randomized online algorithm can achieve a better ratio than $\left(1 - \frac{1}{e}\right) + o(1)$; clearly, this upper bound applies to GENERAL as well.

However, the analysis of RANKING given in [KVV90] was based on a dual-fitting approach, which was recently simplied further by [Vaz21]. We base our study on their work and try to understand and build upon the algorithms and bounds given from an economic viewpoint. [Vaz21] discusses two new ideas, carry over to the analysis of all other versions such as SINGLE-VALUED, k-TYPICAL, GENERAL, SMALL.

Sadly, the question of finding tight algorithms for the GENERAL variant of ADWORDS remain open with a greedy algorithm, which matches each query to the highest bidder, achieves a competitive ratio of 1/2. The current best possible algorithm as given by [HZZ20] marginally improvs over this with a ratio of 0.5016. Followed by this [Vaz21] give an online algorithm for GENERAL. However, it turns out that its structural difficulties do not blend well with their proof technique, and as a result, the revenue generated by the algorithm needs to consist of real as well as “fake” money, failing to ascertain the competitive ratio of their proposed algorithm. We analyze this heuristic theoretically. Following this, we perform experiments with the heuristic to observe any structurally unique and hindering input types.

2 Our Contribution

We first analyze the correctness of arguments proposed in [Vaz21]. Specifically, we analyze the correctness of proofs of competitive analysis of RANKING algorithm for OBM. We also try to understand the deeper significance of the approaches adopted by the author. As we began our analysis in August 2021, we noticed major bugs in the logic of the proof of RANKING, see Section 4.1 for details. Later on we realized that the author themselves updated their manuscript, resolving the bug. Though, the newer version of analysis of RANKING is logically correct, it appears quite analogous to the proof by Dual-Fitting Approach. We then consider the analysis of extension of RANKING to SINGLE-VALUED problem, as discussed in Section 4.1.

Following this, we study the heuristic given for GENERAL problem. We confirm that the partial analysis of the proof is correct. To get insights of the worst case graph inputs and the structural hindrances of the solution, we run experiments on the heuristics with random as well as specifically designed inputs. Our implementation along with all the results are available at `https://github.com/aaysharma/ADWORDS` and the observations and conclusions are discussed in Section 4.2.

3 Preliminaries

3.1 Problem Definitions

Online Bipartite Matching (OBM): Let B be a set of n buyers and S a set of n goods. A bipartite graph $G = (B, S, E)$ is specified on vertex sets B and S, and edge set E, where for $i \in B$, $j \in S$, $(i, j) \in E$ if and only if buyer i *likes* good j. G is assumed to have a perfect matching and therefore each buyer can be given a unique good she likes. Graph G is revealed in the following manner. The n goods are known up-front. On the other hand, the buyers arrive one at a time, and when buyer i arrives, the edges incident at i are revealed.

We are required to design an online algorithm $\mathcal{A}$ in the following sense. At the moment a buyer i arrives, the algorithm needs to match i to one of its unmatched neighbors, if any; if all of i's neighbors are matched, i remains unmatched. The difficulty is that the algorithm does not "know" the edges incident at buyers which will arrive in the future and yet the size of the matching produced by the algorithm will be compared to the best *off-line matching*; the latter of course is a perfect matching.

General Adwords Problem (GENERAL): Let A be a set of m *advertisers*, also called *bidders*, and Q be a set of n *queries*. A bipartite graph $G = (Q, A, E)$ is specified on vertex sets Q and A, and edge set E, where for $i \in Q$ and $j \in A$, $(i, j) \in E$ if and only if bidder j is *interested in* query i. Each query i needs to be matched (Clearly, this is not a matching in the usual sense, since a bidder may be matched to several queries.) to at most one bidder who is interested in it. For each edge (i, j), bidder j *knows* his bid for i, denoted by $\text{bid}(i, j) \in \mathbb{Z}_+$. Each bidder also has a *budget* $B_j \in \mathbb{Z}_+$ which satisfies $B_j \geqslant \text{bid}(i, j)$, for each edge (i, j) incident at j.

Graph G is revealed in the following manner. The m bidders are known up-front and the queries arrive one at a time. When query i arrives, the edges incident at i are revealed, together with the bids associated with these edges. If i gets matched to j, then the matched edge (i, j) is assigned a weight of $\text{bid}(i, j)$. The constraint on j is that the total weight of matched edges incident at it be at most B_j. The objective is to maximize the total weight of all matched edges at all bidders.

Adwords under Single-Valued Bidders (SINGLE-VALUED): SINGLE-VALUED is a special case of GENERAL in which each bidder j will make bids of a single value, $b_j \in \mathbb{Z}_+$, for the queries he is interested in. If i accepts j's bid, then i will be matched to j and the weight of this matched edge will be b_j. Corresponding to each bidder j, we are also given $k_j \in \mathbb{Z}_+$, the maximum number of times j can be matched to queries. The objective is to maximize the total weight of matched edges. Observe that the matching M found in G is a b-matching with the b-value of each query i being 1 and of advertiser j being k_j.

We will define the notion of competitive ratio of a randomized online algorithm in the context of OBM.

Definition 1. *Let $G = (B, S, E)$ be a bipartite graph as specified above. The competitive ratio of a randomized algorithm $\mathcal{A}$ for OBM is defined to be:*

$$c(\mathcal{A}) = \min_{G=(B,S,E)} \min_{\rho(B)} \frac{\mathbb{E}[\mathcal{A}(G, \rho(B))]}{n},$$

where $\mathbb{E}[\mathcal{A}(G, \rho(B))]$ is the expected size of matching produced by $\mathcal{A}$; the expectation is over the random bits used by $\mathcal{A}$. We may assume that the worst case graph and the order of arrival of buyers, given by $\rho(B)$, are chosen by an adversary who knows the algorithm. It is important to note that the algorithm is provided random bits after *the adversary makes its choices.*

3.2 RANKING Algorithm

Algorithm 1 presents an optimal algorithm for OBM. This algorithm picks a random permutation of goods only once. Its competitive ratio is $(1 - \frac{1}{e})$ as shown by previous works. Furthermore, as shown in [KVV90], it is an optimal online bipartite matching algorithm: no randomized algorithm can do better, up to an $o(1)$ term.

Algorithm 1. (Algorithm RANKING)

```
1. Initialization: Pick a random permutation, $\pi$, of the goods in $S$.
2. Online buyer arrival: When a buyer, say $i$, arrives, match her to the first unmatched good she
   likes in the order $\pi$; if none, leave $i$ unmatched.
Output the matching, $M$, found.
```

Algorithm 2. (Algorithm RANKING: Economic Viewpoint)

```
1. Initialization: $\forall j \in S$:   Pick $w_j$ independently and uniformly from $[0,1]$.
   Set price  $p_j \leftarrow e^{w_j - 1}$.
2. Online buyer arrival: When a buyer, say $i$, arrives, match her to the cheapest unmatched good
   she likes; if none, leave $i$ unmatched.
Output the matching, $M$, found.
```

4 Our Results

4.1 RANKING

[Vaz21] remoulded the RANKING algorithm in an economic-viewpoint as depicted in Algorithm 2. It is clear that both the algorithms Algorithm 1 and Algorithm 2 are logically equivalent, so the analysis of one applies to the other.

4.1.1 The Buggy Proof

August 2021 version of [Vaz21] proved RANKING with a rather novel but buggy approach.

For analyzing this algorithm, they define two sets of random variables, u_i for $i \in B$ and r_j, for $j \in S$. These will be called utility of buyer i and revenue of good j, respectively. Each run of RANKING defines these random variables as follows. If RANKING matches buyer i to good j, then define $u_i = 1 - p_j$ and $r_j = p_j$, where p_j is the price of good j in this run of RANKING. Clearly, p_j is also a random variable, which is defined by Step (1) of the algorithm. If i remains unmatched, define $u_i = 0$, and if j remains unmatched, define $r_j = 0$. Observe that for each good j, $p_j \in [\frac{1}{e}, 1]$ and for each buyer i, $u_i \in [0, 1 - \frac{1}{e}]$. Let M be the matching produced by RANKING and let random variable $|M|$ denote its size. Lemma 1 pulls apart the contribution of each matched edge (i,j) into u_i and r_j. Next, they established in Lemma 3 that for each edge (i,j) in the graph, the total expected contribution of u_i and r_j is at least $1 - \frac{1}{e}$. Then, linearity of expectation can be used to reassemble the $2n$ terms in the right hand side of Lemma 1 so they are aligned with a perfect matching in G.

Lemma 1. *[Vaz21, Lemma 6]*

$$\mathbb{E}[|M|] \; = \sum_{i}^{n} \mathbb{E}\left[u_i\right] \; + \; \sum_{j}^{n} \mathbb{E}[r_j].$$

However to analyse the value of $\mathbb{E}[r_j]$, the authors develop the notion of G_j. Assuming that the adversary has picked the order of arrival of buyers, say $\rho(B)$, and Step (1) has been executed. They define several ways of executing Step (2). Let $\mathcal{R}$ denote the run of Step (2) on the entire graph G.

Corresponding to each edge $e = (i,\ j)$, let G_e denote graph G with edge e removed. Define $\mathcal{R}_e$ to be the run of Step (2) on graph G_e. . They define the threshold utility value u_e as follows:

Definition 2. *Let $e = (i, j) \in E$ be an arbitrary edge in G. Define random variable, u_e, called the* threshold *for edge e, to be the utility of buyer i in run $\mathcal{R}_e$.*

The author then proves the following lemma, which is quite analgous to the proofs from Dual-Fitting:

Lemma 2. *[Vaz21, Lemma 9, Faulty Version] Corresponding to each edge $(i, j) \in E$, the following hold.*

1. *$u_i \geqslant u_e$, where u_i and u_e are the utilities of buyer i in runs $\mathcal{R}$ and $\mathcal{R}_e$, respectively.*
2. *If $p_j < 1 - u_e$, then j will definitely be matched in run $\mathcal{R}$.*

The intuitive reason for the next, and most crucial, lemma is the following. The smaller u_e is, the larger is the range of values for p_j, namely $[0, 1 - u_e)$, over which (i, j) will be matched and j will accrue revenue of p_j. Integrating p_j over this range, and adding $\mathbb{E}[u_i]$ to it, gives the desired bound. Crucial to this argument is the fact that p_j is independent of u_e. This follows from the fact that u_e is determined by run $\mathcal{R}_j$ on graph G_j, which does not contain vertex j.

We next point to the main result which holds false in the faulty version of the paper:

Lemma 3. *Corresponding to each edge $(i, j) \in E$,*

$$\mathbb{E}[u_i + r_j] \geqslant 1 - \frac{1}{e}.$$

Proof. By the first part of Lemma 2, $\mathbb{E}[u_i] \geqslant \mathbb{E}[u_e]$.

Next, they try to lower bound $\mathbb{E}[r_j]$. Let $z \in [0, 1 - \frac{1}{e}]$ conditioning on the event $u_e = z$. Here the author assume that u_e is independent of p_j. We claim that this is wrong. A possible counter intuition to this is that observation is that u_e is determined by the run $\mathcal{R}_e$. This is conducted on graph G_e. Although this graph G_e does not contain edge $e = (i,\ j)$, it still contains both the end vertices i and j. The rest of the graph still has edges to the two vertices of e, which does in fact affect the value of p_j in the original run $\mathcal{R}$.

When $p_j < 1 - z$, the random variable r_j is defined as follows: pick x uniformly at random from $[0, w)$ and let r_j be e^{x-1} We again follow from the last argument that although p_j can be seen as obtained by picking x uniformly at random from the interval $[0, 1]$ and outputting e^{x-1}. It is not established as to how the distribution of p_j given in this interval is uniform conditioned on u_e. Only if the author could somehow establish the independence of u_e and p_j, the former could have been true. □

4.1.2 Corrected Proof: RANKING and SINGLE-VALUED

The author rectified over their error by redefining the run $\mathcal{R}_j$ as follows: While running Algorithm 2, assume that the adversary has picked the order of arrival of buyers, say $\rho(B)$, and Step (1) has been executed. We next define several ways of executing Step (2). Let $\mathcal{R}$ denote the run of Step (2) on the entire graph G. Corresponding to each good j, let G_j denote graph G with vertex j removed. Define $\mathcal{R}_\mathrm{j}$ to be the run of Step (2) on graph G_j. This gives the following new definition of threshold variable:

Definition 3. *Let $e = (i, j) \in E$ be an arbitrary edge in G. Define random variable, u_e, called the* threshold *for edge e, to be the utility of buyer i in run $\mathcal{R}_j$. Clearly, $u_e \in [0, 1 - \frac{1}{e}]$.*

```
Algorithm 3. ($\mathcal{A}_3$: Algorithm for GENERAL)

  1. Initialization: $M \leftarrow \emptyset,\ W \leftarrow 0$ and $W_f \leftarrow 0$
     $\forall j \in A$, do:
       (a) Pick $w_j$ uniformly from $[0,1]$ and set price $p_j \leftarrow e^{w_j - 1}$.
       (b) $r_j \leftarrow 0$.
       (c) $L_j \leftarrow B_j$.

  2. Query arrival: When query $i$ arrives, do:
       (a) $\forall j \in A$ s.t. $(i,j) \in E$ and $L_j > 0$ do:
             i. $\text{ebid}(i,j) \leftarrow \text{bid}(i,j) \cdot (1 - p_j)$.
            ii. Offer effective bid of $\text{ebid}(i,j)$ to $i$.
       (b) Query $i$ accepts the bidder whose effective bid is the largest.
           (If there are no bids, matching $M$ remains unchanged.)
           If $i$ accepts $j$'s bid, then do:
             i. Set utility: $u_i \leftarrow \text{bid}(i,j) \cdot (1 - p_j)$.
            ii. Update revenue: $r_j \leftarrow r_j + \text{bid}(i,j) \cdot p_j$.
           iii. Update matching: $M \leftarrow M \cup (i,j)$.
            iv. Update weight: $W \leftarrow \min\{L_i, \text{bid}(i,j)\}$ and $W_f \leftarrow \max\{0,\ \text{bid}(i,j) - L_i\}$.
             v. Update $L_j$: $L_j \leftarrow L_j - \min\{L_j, \text{bid}(i,j)\}$.
  3. Output: Output matching $M$, real money spent $W$, and fake money spent $W_f$.
```

Crucial to this argument is the fact that p_j is independent of u_e. This follows from the fact that u_e is determined by run $\mathcal{R}_j$ on graph G_j, which does not contain vertex j. We further verified the correctness of the proof. However, the proof seems to be just a combinatorial rephrasing of dual-fitting approach.

We then analyzed the proof of same competitive ratio for the SINGLE-VALUED Variant given by [Vaz21, Algorithm 19]. The proof is correct and follows by a simple generalization of exact same proof techniques from RANKING. By simply replacing the notion of a matching edge with that of a j-star, we can understand the notion of the proof.

4.2 GENERAL

Following this, we study the heuristic for GENERAL problem given in [Vaz21]. However, it turns out that its structural difficulties do not blend well with their proof technique, and as a result, the revenue generated by the algorithm needs to consist of real as well as "fake" money, failing to ascertain the competitive ratio of their proposed algorithm. We analyze this heuristic theoretically. Following this, we perform experiments by running the heuristic on random as well as specially designed bipartite graphs. A summary of our observations is shown in Table 1.

We observe from the table that for most of the graphs, irrespective of the size of the input, the heuristic performs well over $1 - \frac{1}{e} \simeq 0.632121$. In fact, for runs 10 and 11, the algorithm performs as good as the optimal. However, for run 15, we are able to observe a competitive ratio of $0.57732 < 0.632121$. With this observation, the algorithm does not seem quite hopeful. However, run 15 is the only anomaly observed in a multitude of experimental runs over random inputs, giving the slightest hope for the expected values. For future, our aim will be to analyse the struture of graphs similar to run 15 and prove an expected lower bound for the heuristic by minimizing the expected value of wasted fake money.

Our Codes and outputs of all runs are available at `https://github.com/aaysharma/ADWORDS`

#	Advertisers (m)	Queries (n)	W	W_f	Optimal	Competitive Ratio
1	15	6	186	26	212	0.834081
2	5	15	343	0	353	0.971671
3	20	4	128	17	149	0.85906
4	5	7	88	8	109	0.807339
5	4	9	128	6	145	0.882759
6	10	5	76	11	92	0.826087
7	9	9	138	12	164	0.841463
8	8	10	159	0	175	0.908571
9	9	5	72	11	88	0.818182
10	4	2	36	0	36	1
11	6	3	58	0	58	1
12	10	5	67	17	92	0.728216
13	7	10	136	11	174	0.7816
14	8	6	80	28	113	0.708
15	5	6	56	12	97	0.57732

Table 1: Table lists the run of Algorithm for GENERAL with varying values of number of advertizers (m) and queries (n).

References

[HZZ20] Zhiyi Huang, Qiankun Zhang, and Yuhao Zhang. Adwords in a panorama. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1416–1426. IEEE, 2020.

[KVV90] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.

[MSVV07] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5), 2007.

[Vaz21] Vijay V. Vazirani. Randomized, budget-oblivious online algorithms for adwords, 2021.