Which Mechanism for Sponsored Search Auctions with Externalities?

Nicola Gatti
Politecnico di Milano
Piazza Leonardo da Vinci 32
Milano, Italy
ngatti@elet.polimi.it

Marco Rocco
Politecnico di Milano
Piazza Leonardo da Vinci 32
Milano, Italy
mrocco@elet.polimi.it

ABSTRACT

Sponsored search is one of the most successful applications of economic mechanisms in real life. A crucial issue is the modeling of the user behavior to provide the best targeting of ads to each user. Experimental studies show that the *click through rate* of an ad is dramatically affected by both its position and the other displayed ads. However, these externalities rise severe currently open computational issues in the determination of the best allocation and of the payments, preventing their adoption in practice so far. In the present paper, we provide a number of results when the most famous externality model, the *cascade model*, is adopted: we design the first exact algorithm for computing the efficient allocation, we show that the previously presented constant–approximation algorithm does not lead to any incentive compatible mechanism, we design a monotonic constant–approximation algorithm for finding the allocation and two different polynomial–time algorithms for the payments, each with different properties, leading to incentive compatible mechanisms. Finally, we provide a thorough experimental evaluation of the presented algorithms with *Yahoo! Webscope A3 dataset* to identify which mechanism should be adopted in concrete applications.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Multi–agent systems

General Terms

Algorithms, Economics

Keywords

Auction and mechanism design

1. INTRODUCTION

In sponsored search auctions a publisher selects ads to be placed in a number of slots on a web page and an advertiser pays the publisher only when its ad is clicked. Sponsored search auctions play a central role in Internet monetization, e.g., in the first half of 2010, revenue from online advertising totalled $12.1 billion in the U.S. alone, of which search revenue accounted for 47%, dominating display ads, the second–largest revenue source [10].

The auction model currently most adopted is the *generalized second price* (GSP) [6], in which ads are allocated to slots in decreasing order of value and an advertiser pays the next highest value when clicked. GSP auction has been demonstrated not to be *truthful*, i.e., bidding true values is not the best strategy of the advertisers, and this poses severe issues on the stability of the market [6]. Indeed, while with complete information the GSP admits multiple Nash equilibria in pure strategies and the worst (in terms of revenue) equilibrium for the mechanism corresponds to the truthful equilibrium obtained with *Vickrey–Clarke–Groves* (VCG) mechanism [14], when, as usual, information is uncertain the GSP can admit Bayes–Nash equilibria that are much worse (up to $\frac{1}{8}$) for the mechanism w.r.t. the VCG equilibrium [13]. Thus, although the VCG is not used by the main search engines (but it is by Facebook), its study is commonly considered of extraordinary importance.

A crucial issue in sponsored search auctions is the study of effective user models and their exploitation in the auction mechanism. Recently, a number of works showed that *externalities* play an important role in the user behavior [1, 7, 12]. On the other hand, externalities can make the winner determination problem intractable, even when approximated, e.g., it is $\mathcal{APX}$–hard in [7]. The most famous user model is the *cascade model* [12], in which a user is assumed to scan the ads sequentially from the slot in the top to the slot in the bottom with a probability to observe the subsequent slot that depends on the positions (*position–dependent externality*) and on the ads (*ad–dependent externality*). Although several experimental activities confirmed that the behavior of the users is close to the one described by the cascade model [5, 11], the adoption of this model in practice is currently an open problem. Indeed, it is not known whether it is possible to design a *truthful economic mechanism* in which the problems to find the (approximate) optimal allocation and the payments are in $\mathcal{P}$ and, even if they are in $\mathcal{P}$, whether their compute time is short enough for the adoption of the mechanism in online situations.

In this paper we explore the problem of designing computationally efficient truthful mechanisms for sponsored search auctions with externality. The main contributions we provide are as follows.

- We develop a branch–and–bound algorithm to compute the efficient allocation whose complexity is exponential in the number of slots.
- We show that the polynomial time allocation function

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

with $\frac{1-\epsilon}{4}$ approximation ratio designed in [12] is not *monotone* [3] and therefore that no mechanism adopting such allocation function can be truthful.

- We design a monotone polynomial–time allocation function providing an approximation of $\frac{1-\epsilon}{4}$ of the efficient allocation.

- We design two polynomial–time payment functions, each with a different computational complexity. We use them together with our allocation function to design computationally efficient truthful mechanisms, each with different properties. From the slowest to the fastest: the first mechanism is truthful in expectation and individually rational and weakly budget balanced in *ex post*, and the second mechanism is truthful and weakly budget balanced in expectation and individually rational in *ex post*.

- For the two proposed approximation mechanisms we derive theoretical bounds on the loss in probability of the mechanism w.r.t. the average behavior.

- We provide a thorough experimental evaluation of the algorithms exploiting the *Yahoo! Webscope A3 dataset* with the aim to determine the best mechanism in terms of tradeoff between compute time and quality of the allocation for each combination of ads and slots.

2. PROBLEM STATEMENT

2.1 Sponsored search auction model

The sponsored search auction model without externalities is composed of the following ingredients:

- $N = \{a_1, \ldots, a_n, a_\perp\}$ is the set of ads and $a_\perp$ is a fictitious ad — w.l.o.g. we assume each advertiser (agent) to have a single ad, so each agent i can be identified with ad a_i;

- $K = \{s_1, \ldots, s_k, s_\perp\}$ is the set of slots ordered from the top to the bottom and $s_\perp$ is a fictitious slot;

- $q_{a_i} \in [0,1]$ is the *quality* of ad a_i (i.e., the probability a user will click ad a_i when observed);

- $v_{a_i} \in V_{a_i} \subseteq \mathbb{R}^+$ is the value for agent i when ad a_i is clicked by a user — $\mathbf{v} = (v_{a_1}, \ldots, v_{a_k})$ is the value profile;

- $\hat{v}_{a_i} \in V_{a_i}$ is the value reported by agent i — $\hat{\mathbf{v}} = (\hat{v}_{a_1}, \ldots, \hat{v}_{a_k})$ is the reported value profile;

- Θ is the set of ordered allocations of ads to slots, where each ad cannot be allocated in more than one slot — we assign $a_\perp$ to s_j to denote that no ad is displayed in s_j, and we assign a_i to $s_\perp$ to denote that a_i is not displayed.

With a slight abuse of notation we let:

- $\theta(a_i)$ to denote the slot in which a_i is allocated in θ;

- $\theta(s_j)$ to denote the ad allocated in s_j in θ.

The externalities introduced by the cascade model assume the user to have a Markovian behavior, starting to observe the slots from the first (i.e., s_1) to the last (i.e., s_k) where the transition probability from slot s_j to slot s_{j+1} is given by the product of two parameters:

- (*ad–dependent externalities*) $c_{a_i} \in [0,1]$ is the *continuation probability* of a_i — it is assumed $c_{a_\perp} = 1$;

- (*position–dependent externalities*) $\overline{\lambda}_{s_j} \in [0,1]$ is the *factorized prominence* of s_j — it is assumed $\overline{\lambda}_{s_\perp} = 0$.

The click through rate $CTR_{a_i}(\theta) \in [0,1]$ of ad a_i in an allocation θ is the probability a user will click a_i and it is formally defined as $CTR_{a_i}(\theta) = \lambda_{\theta(a_i)} \cdot C_{a_i}(\theta) \cdot q_{a_i}$ where $C_{a_i}(\theta) = \prod_{a_j \in N : \theta(a_j) < \theta(a_i)} c_{a_j}$ and $\lambda_{\theta(a_i)} = \prod_{j \in K : j \leq \theta(a_i)} \overline{\lambda}_j$. Parameter λ_{s_j} is commonly called *prominence* [12].

2.2 Computational mechanism design problem

The problem we study is the design of an economic mechanism $\mathcal{M}$, composed of

- an *allocation function* $f : \times_{i \in N} V_{a_i} \to \Theta$ and

- a *payment function* $p_{a_i} : \times_{i \in N} V_{a_i} \to \mathbb{R}$.

Each agent i has a linear utility $u_i(v_{a_i}, \hat{\mathbf{v}}) = v_{a_i} \cdot CTR_{a_i}(f(\hat{\mathbf{v}})) - p_{a_i}(\hat{\mathbf{v}})$ in expectation over the clicks. Agents pay only when clicked and the payment is $\frac{p_{a_i}(\hat{\mathbf{v}})}{CTR_{a_i}(f(\hat{\mathbf{v}}))}$. We are interested in mechanisms satisfying the following properties.

Definition 2.1. *Mechanism $\mathcal{M}$ is dominant strategy incentive compatible (DSIC) if reporting true values is a dominant strategy for every agent (i.e., $\hat{v}_{a_i} = v_{a_i}$).*

Definition 2.2. *Mechanism $\mathcal{M}$ is individually rational (IR) if no agent acting truthfully prefers to abstain from participating to the mechanism rather than participating.*

Definition 2.3. *Mechanism $\mathcal{M}$ is weakly budget balance (WBB) if the mechanism is never in deficit.*

Definition 2.4. *Mechanism $\mathcal{M}$ is computationally tractable when both f and p are computable in polynomial time.*

We introduce an additional property that is necessary and sufficient to have incentive compatibility in dominant strategies when the problem is linear [3], as ours is.

Definition 2.5. *Allocation function f is monotone when: $CTR_a(f(\hat{v}_{a_1}, \ldots, \hat{v}_a, \ldots, \hat{v}_{a_n}))$ monotonically increases in $\hat{v}_a$ for any ad a [3].*

A special kind of monotone allocation functions are the efficient ones:

Definition 2.6. *Mechanism $\mathcal{M}$ is allocatively efficient if $f = \arg\max_\theta \sum_{a_i \in N} v_{a_i} \cdot CTR_{a_i}(f(\mathbf{v}))$.*

2.3 Known results and open problems

The application of the standard VCG mechanism to the above problem presents severe computational issues. Without either ad–dependent and/or position–dependent externalities, f and p are computationally easy, as shown in [12]. With both externalities, although there is no proof showing

the allocation problem to be $\mathcal{NP}$–hard, this problem is commonly considered intractable.[1] It is worth noting that, in the worst case, an algorithm enumerating all the possible allocations has an asymptotical complexity of $O(n^k)$. This is because in no optimal allocation $a_\perp$ is associated with some slot, as shown in [12]. Thus, when k is bounded, as it is usual in practice, the problem is polynomial, but it could be unaffordable in real time for large values of k and n. However, no work in the literature deals with the problem to find an efficient allocation and to determine the range of values of the parameters such that such allocation can be found in real time in online applications.

In the literature, the problem of designing a non–efficient computationally tractable mechanism has been partially explored. In [12], the authors present an allocation function that guarantees a social welfare non–smaller than $\frac{1}{4}$ of the efficient allocation, whose computational complexity is pseudopolynomial. The authors strengthen the result, providing an algorithm with approximation ratio of $\frac{1-\epsilon}{4}$, running with $O(\frac{1}{\epsilon})$, where $\epsilon > 0$, and polynomial in the other terms. The authors pose the question whether it is possible to design computationally efficient payments such that the resulting mechanism is incentive compatible. In this paper, we show that such a question has a negative answer even when payments require exponential time and this is because the allocation function induced by the algorithm in [12] is not monotone. Hence, no monotone polynomial–time allocation function with strictly positive approximation ratio is currently known in literature.

3. VCG MECHANISM

We propose a branch–and–bound algorithm for the efficient allocation function f_E, necessary for the application of the VCG mechanism. It is reported in Algorithm 1. Basically, it is a recursive backtracking algorithm. Initially, the algorithm is called with $(\theta_0, 1, 0)$ where in θ_0 all the slots are associated with $a_\perp$. Then, at each call, the algorithm at Step 3 adds a new ad a_i to the partial current allocation θ unless all the slots are assigned (Step 2), checks at Step 5, by using an admissible heuristic $h(\theta)$, whether the new partial allocation can lead to an allocation strictly better than the best allocation θ^* (whose value is best) found so far and, in the affirmative case, the algorithm is recursively called from the new allocation θ (Step 6). We define $h(\theta)$ as the minimum between:

- the optimal value given by the remaining slots when $c_{a_i} = 1$ for all a_i that are not allocated yet,
- the optimal value given by the remaining slots when for all λ_{s_j}, $\lambda_{s_j} = \lambda_{s_l}$, where s_l is the last allocated slot in θ.

Both values are computable in polynomial time. Heuristic $h(\theta)$ obviously provides an overestimation of the optimal value of the remaining allocation of θ and therefore the algorithm is sound. The algorithm runs with $O(n^k)$. The VCG payments require the determination of the best allocation in k subproblems, as described in [14]. Therefore the complexity of the VCG mechanism is $O(k \cdot n^k)$.

[1]The hardness proof used in [7] cannot be directly applied to the cascade model because the reduction assumes λ to be equal to one and the cascade model is easy in this case.

Algorithm 1 f_EAllocation$(\theta, j, \mathsf{best})$

```
1: θ* ← θ
2: if j ≤ k then
3:    for all a_i ≠ θ(s_z) with z < j do
4:       θ(s_j) ← a_i
5:       if Σ_h CTR_{a_h}(θ)·v_{a_h} + C_{θ(s_j)}(θ)·h(θ) > best then
6:          (θ', best') ← f_E Allocation(θ, j+1, best)
7:          if best' > best then
8:             best ← best'
9:             θ* ← θ'
10: return (θ*, best)
```

4. NON–EFFICIENT ALLOCATION FUNCTIONS

4.1 Non–monotonicity of the $\frac{1-\epsilon}{4}$ [13]

We briefly review the pseudopolynomial–time $\frac{1}{4}$–approximation algorithm from which the authors derived their $\frac{1-\epsilon}{4}$ approximation algorithm in [12] because we can show that even the allocation function induced by this algorithm is not monotone. The $\frac{1}{4}$–approximation algorithm works as follows:

- find the allocation θ maximizing $\sum_{a_i} \lambda_{\theta(a_i)} \cdot v_{a_i} \cdot q_{a_i}$,
- under the constraint that, letting s_l the last slot with an ad different from $a_\perp$, $C_{\theta(s_l)}(\theta) \geq \frac{1}{2}$.

The last constraint can be also represented as $\sum_{i=1}^{l} \log_2 \frac{1}{c_{a_i}} \leq 1$.

The property of monotonicity of an allocation function can be easily captured in the case of the sponsored search auction with externalities observing Fig. 1: increasing v_{a_i} and keeping fix all the others $v_{a_{-i}}$ the allocation θ changes and the $CTR_{a_i}(\theta)$ monotonically increases. We state the following.

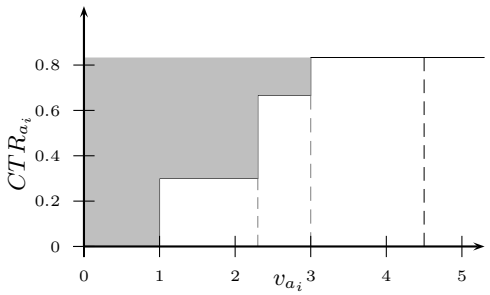

Figure 1: Relation between CTR_{a_i} and v_{a_i}.

PROPOSITION 4.1. *The allocation function induced by the above $\frac{1}{4}$–approximation algorithm is not monotone.*

Proof. Consider an auction with 5 ads and 4 slots, where:

	q_{a_i}	v_{a_i}	c_{a_i}
a_1	0.500	2.0	0.50
a_2	0.300	3.0	0.90
a_3	0.250	2.0	0.90
a_4	0.200	x	1.00
a_5	0.111	10.0	0.10

	λ
s_1	1.00
s_2	0.60
s_3	0.50
s_4	0.36

We can show that the allocation function is not monotone in v_{a_4}. The allocations chosen by the allocation function in the range $x \in [1.554, 2.500]$ are:

- $1.554 \leq x \leq 2.23$: the allocation is $\{a_2, a_3, a_4, a_5\}$ and $CTR_{a_4} = \lambda_{s_3} \cdot c_{a_2} \cdot c_{a_3} \cdot q_{a_4} = 0.081$,
- $2.23 < x \leq 2.5$: the allocation is $\{a_1, a_4, a_5\}$ and $CTR_{a_4} = \lambda_{s_2} \cdot c_{a_1} \cdot q_{a_4} = 0.060$.

Therefore, increasing $v_{a_4} = x$ along $[1.554, 2.5]$, the CTR_{a_4} decreases from 0.081 at $x \leq 2.23$ to 0.06 at $x > 2.23$. This shows that the allocation function is not monotone. □

As shown in [3], with linear utility functions, the monotonicity of the allocation function is necessary to have incentive compatibility. Thus, no incentive compatible mechanism can be designed when the above allocation function is used. The same holds for the $\frac{1-\epsilon}{4}$–approximation algorithm described in [12], given that an example similar to the above can produced also for this allocation function.

4.2 Pseudopolynomial–time monotone $f_{\frac{1}{4}}$

We propose a variation of the $\frac{1}{4}$–approximation algorithm described in [12] that we prove to be monotone and to provide an approximation of $\frac{1}{4}$. This approximation algorithm runs in exponential time when numeric precision is exact and it is pseudopolynomial when finite precision is accepted. We call this allocation function $f_{\frac{1}{4}}$. It works as follows:

- find the allocation θ maximizing $\sum_{a_i} CTR_{a_i}(\theta) \cdot v_{a_i}$,
- under the constraints that, letting s_l the last slot with an ad different from $a_\perp$, $C_{\theta(s_l)}(\theta) \geq \frac{1}{2}$, and
- under the constraints that for every $1 < j < l$ it holds $q_{\theta(s_j)} \cdot v_{\theta(s_j)} \leq q_{\theta(s_{j-1})} \cdot v_{\theta(s_{j-1})}$.

The algorithm is reported in Algorithm 2. For each ad $\overline{a} \in N$, the algorithm removes $\overline{a}$ from N (Step 1) and at Step 2 sorts all the remaining ads in N in decreasing order in $q_{a_i} \cdot v_{a_i}$. Then it searches for the best allocation in which $\overline{a}$ is the last displayed ad (not necessarily in slot s_k) under the two above constraints. This is done by dynamic programming in a similar fashion used by the dynamic programming algorithm for the knapsack problem with cardinality and capacity constraints [15]. We use a data structure list in which we insert partial allocations characterized by (ϕ, ψ, χ), where ϕ is the value of the allocation, ψ is the number of allocated slots, χ is the product of the continuations probabilities of all the ads in the allocation. For each ad $a_j \in N \setminus \{\overline{a}\}$, at Step 4 a partial allocation in which a_j is displayed in the first slot s_1 is added to $\mathsf{list}(\overline{a}, a_j)$, and for all the partial allocations in $\mathsf{list}(\overline{a}, \mathsf{prev}(a_j))$, where $\mathsf{prev}(a_j)$ is the ad that precedes a_j in the ordered set $N \setminus \{\overline{a}\}$, a new partial allocation is added to $\mathsf{list}(\overline{a}, a_j)$ in which a_j is appended (Steps $5-6$). Then, at Step 7 all the dominated partial allocations of $\mathsf{list}(\overline{a}, a_j)$ are removed. Exactly, $(\phi, \psi, \chi) \in \mathsf{list}(\overline{a}, a_j)$ is dominated when:

- $\chi < \frac{1}{2}$,
- $\psi = k$,
- there exists $(\phi', \psi', \chi') \in \mathsf{list}(\overline{a}, a_j)$ with $\phi' \geq \phi$, $\psi' \leq \psi$, $\chi' \geq \chi$.

At Steps $9-10$, $\overline{a}$ is added to all the partial allocations in $\mathsf{list}(\overline{a}, \mathsf{last}(N \setminus \{\overline{a}\}))$, where last returns the last element of the ordered set, updating ϕ with the value of $\overline{a}$. In this way, all the allocations satisfying the two above constraints are generated. Finally, the best allocation is chosen among all those in $\mathsf{list}(\overline{a}, \mathsf{last}(N \setminus \{\overline{a}\}))$ for every $\overline{a}$ (Step 11).

Algorithm 2 $f_{\frac{1}{4}}\mathsf{Allocation}$

1: **for** all $\overline{a} \in N$ **do**
2: sort $N \setminus \{\overline{a}\}$ in decreasing order in $q_{a_j} \cdot v_{a_j}$
3: **for** all $a_j \in N \setminus \{\overline{a}\}$ **do**
4: add $(\lambda_{s_1} \cdot v_{a_j} \cdot q_{a_j}, 1, c_{a_j})$ to $\mathsf{list}(\overline{a}, a_j)$
5: **for** all $(\phi, \psi, \chi) \in \mathsf{list}(\overline{a}, \mathsf{prev}(a_j))$ **do**
6: add $(\phi + \lambda_{s_{\psi+1}} \cdot \chi \cdot v_{a_j} \cdot q_{a_j}, \psi + 1, \chi \cdot c_{a_j})$ to $\mathsf{list}(\overline{a}, a_j)$
7: remove dominated (ϕ, ψ, χ) in $\mathsf{list}(\overline{a}, a_j)$
8: add $(\lambda_{s_1} \cdot v_{\overline{a}} \cdot q_{\overline{a}}, 1, c_{\overline{a}})$ to $\mathsf{list}(\overline{a}, \overline{a})$
9: **for** all $(\phi, \psi, \chi) \in \mathsf{list}(\overline{a}, \mathsf{last}(N \setminus \{\overline{a}\}))$ **do**
10: add $(\phi + \lambda_{s_{\psi+1}} \cdot \chi \cdot v_{\overline{a}} \cdot q_{\overline{a}}, \psi + 1, \chi \cdot c_{\overline{a}})$ to $\mathsf{list}(\overline{a}, \overline{a})$
11: **return** $(\phi, \psi, \chi) \in \mathsf{list}(\overline{a}, \overline{a})$ maximizing ϕ for $\overline{a} \in N$

PROPOSITION 4.2. *The asymptotical computational complexity of Algorithm 2 is exponential.*

Proof. The worst case complexity corresponds to the largest number of elements in list. In order to find this number, it is necessary to make all the parameters $c_{a_i}, \lambda_{s_j}, v_{a_i}, q_{a_i}$ integer. This is done by multiplying all the parameters by a constant $O(10^L)$ where L is the used bit length. Focus on ϕ and χ: the latter clearly requires a shorter bit length. However, χ requires a bit length $O(L^k)$. Thus, the size of $\mathsf{list}(\overline{a}, a_j)$ is $O(k \cdot n^2 \cdot L^k)$. □

PROPOSITION 4.3. *When finite precision is accepted, the asymptotical computational complexity of Algorithm 2 is* $O(n^2 \cdot k \cdot 10^L)$ *where L is the bit length used to represent* $\log_2$.

Proof. We compute each $\log_2(\frac{1}{c_{a_i}})$ for each a_i and we make them integer by multiplying them by 10^ω for some ω. The capacity constraint $C_{a_i} \geq \frac{1}{2}$ can be written as $\sum_{a_i} \log_2(\frac{1}{c_{a_i}}) \leq 1$. Thus the number of possible different values for χ in Algorithm 2 is $O(10^L)$. Therefore, by removing dominated elements in list, the maximum size is $O(n^2 \cdot k \cdot 10^L)$. □

Thus, the above algorithm runs in pseudopolynomial time, being exponential in the numeric representation.

PROPOSITION 4.4. *The allocation function induced by Algorithm 2 is monotone.*

Proof. Basically, the proof is because the allocation function is maximal (efficient) in its range. Assume by contradiction that the allocation function is not monotone and therefore that there are two allocations $\overline{\theta}, \tilde{\theta}$ such that:

- $\overline{\theta} = f_{\frac{1}{4}}(\overline{v}_{a_i}, v_{a_{-i}})$
- $\tilde{\theta} = f_{\frac{1}{4}}(\tilde{v}_{a_i}, v_{a_{-i}})$
- $CTR_{a_i}(\tilde{\theta}) > CTR_{a_i}(\overline{\theta})$
- $\overline{v}_{a_i} > \tilde{v}_{a_i}$

The cumulative value of $\overline{\theta}$ can be expressed as $CTR_{a_i}(\overline{\theta}) \cdot v_{a_i} + \overline{\gamma}$, where $\overline{\gamma}$ is a constant. Similarly, the cumulative value of $\tilde{\theta}$ can be expressed as $CTR_{a_i}(\tilde{\theta}) \cdot v_{a_i} + \tilde{\gamma}$, where $\tilde{\gamma}$ is constant. From the first two above conditions:

- $CTR_{a_i}(\overline{\theta}) \cdot \overline{v}_{a_i} + \overline{\gamma} \geq CTR_{a_i}(\tilde{\theta}) \cdot \overline{v}_{a_i} + \tilde{\gamma}$

- $CTR_{a_i}(\tilde{\theta}) \cdot \tilde{v}_{a_i} + \tilde{\gamma} \geq CTR_{a_i}(\overline{\theta}) \cdot \tilde{v}_{a_i} + \overline{\gamma}$

Combining these two inequalities with $CTR_{a_i}(\tilde{\theta}) > CTR_{a_i}(\overline{\theta})$, it follows $\tilde{v}_{a_i} \geq \overline{v}_{a_i}$ that is in contradiction with $\overline{v}_{a_i} > \tilde{v}_{a_i}$. Therefore, $f_{\frac{1}{4}}$ is monotone. □

PROPOSITION 4.5. *Algorithm 2 is a $\frac{1}{4}$–approximation algorithm of the efficient allocation.*

Proof. Algorithm 2 finds the efficient allocation among a subset of allocations $\overline{\Theta} \subseteq \Theta$ determined by the two constraints defined above. The allocation returned by the algorithm presented in [12] belongs to $\overline{\Theta}$ and it is a $\frac{1}{4}$–approximation of the efficient allocation. Given that such an algorithm does not return the optimal allocation in the range $\overline{\Theta}$, optimizing a different objective function from the social welfare, the allocation function we present above cannot find a worse allocation. Therefore, $f_{\frac{1}{4}}$ is at least a $\frac{1}{4}$–approximation algorithm. □

4.3 Polynomial–time monotone $f^c_{\frac{1-\epsilon}{4}}$

It is possible to produce a polynomial time $1-\epsilon$ approximation algorithm of the one presented in the previous section, obtaining thus an $\frac{1-\epsilon}{4}$–approximation algorithm for the efficient allocation f_E. We call such allocation function $f^c_{\frac{1-\epsilon}{4}}$. To obtain this, we round the continuation probabilities as $\left\lfloor \frac{\log_2(\frac{1}{c})}{\tau} \right\rfloor$, where τ depends on ϵ. We have modified $f_{\frac{1}{4}}$ in the following way:

- the third component χ of (ϕ, ψ, χ) is $\sum \left\lfloor \frac{\log_2(\frac{1}{c_{a_i}})}{\tau} \right\rfloor$ over the allocated a_i,

- the capacity constraint $C_{a_i} \geq \frac{1}{2}$ is substituted with $\sum_{a_i} \left\lfloor \frac{\log_2(\frac{1}{c_{a_i}})}{\tau} \right\rfloor \leq \frac{1}{\tau}$.

PROPOSITION 4.6. *Allocation function $f^c_{\frac{1-\epsilon}{4}}$ returns a $\frac{1-\epsilon}{4}$–approximation of the efficient allocation.*

Proof. We provide a constructive proof, determining τ to have that $f^c_{\frac{1-\epsilon}{4}}$ is an $1-\epsilon$ approximation of $f_{\frac{1}{4}}$. By definition, we have $\tau \left\lfloor \frac{\log_2(\frac{1}{c})}{\tau} \right\rfloor \leq \log_2(\frac{1}{c}) \leq \tau \left(\left\lfloor \frac{\log_2(\frac{1}{c})}{\tau} \right\rfloor + 1 \right)$. We let:

- $\theta^* = f_{\frac{1}{4}}$

- $\theta^*_\epsilon = f^c_{\frac{1-\epsilon}{4}}$

We can write:

$$\sum_i \lambda_{\theta^*_\epsilon(a_i)} \cdot C_{a_i}(\theta^*_\epsilon) \cdot v_{a_i} \cdot q_{a_i} \geq$$

$$\sum_i \lambda_{\theta^*_\epsilon(a_i)} \cdot \left(\prod_{\theta^*_\epsilon(a_j) < \theta^*_\epsilon(a_i)} 2^{-\tau\left(\left\lfloor \frac{\log_2(\frac{1}{c_{a_j}})}{\tau} \right\rfloor + 1\right)} \right) v_{a_i} \cdot q_{a_i} \geq$$

$$\sum_i \lambda_{\theta^*(a_i)} \cdot \left(\prod_{\theta^*(a_j) < \theta^*(a_i)} 2^{-\tau\left(\left\lfloor \frac{\log_2(\frac{1}{c_{a_j}})}{\tau} \right\rfloor + 1\right)} \right) v_{a_i} \cdot q_{a_i} \geq$$

$$\sum_i \lambda_{\theta^*(a_i)} \cdot \left(\prod_{\theta^*(a_j) < \theta^*(a_i)} 2^{\log_2(c_{a_j}) - \tau} \right) v_{a_i} \cdot q_{a_i} \geq$$

$$2^{-k\tau} \cdot \sum_i \lambda_{\theta^*(a_i)} \cdot C_{a_i}(\theta^*) \cdot v_{a_i} \cdot q_{a_i} =$$

$$(1-\epsilon) \cdot \sum_i \lambda_{\theta^*(a_i)} \cdot C_{a_i}(\theta^*) \cdot v_{a_i} \cdot q_{a_i}$$

Thus, we obtain: $\tau = \frac{\log_2(\frac{1}{1-\epsilon})}{k}$.. Given the value of τ and that the constraint $\sum_{a_i} \left\lfloor \frac{\log_2(\frac{1}{c})}{\tau} \right\rfloor \leq \frac{1}{\tau}$ is weaker than $C_{a_i} \geq \frac{1}{2}$, no potential optimal solution is discarded due to rounding. □

We focus on the computational complexity.

PROPOSITION 4.7. *The asymptotical computational complexity of $f^c_{\frac{1-\epsilon}{4}}$ is $O(n^2 \cdot k^2 \cdot \frac{1}{\epsilon})$.*

Proof. In this case, the maximum number of elements in $\mathsf{list}(\overline{a}, a_j)$ is $O(k \cdot \frac{1}{\tau})$ thanks to dominations. This means that $\mathsf{list}(\cdot,\cdot)$ contains no more than $O(n^2 \cdot k \cdot \frac{1}{\tau})$ elements. Given that $\log_2(\frac{1}{1-\epsilon}) \to \epsilon$ as $\epsilon \to 0$, the complexity is $O(n^2 \cdot k^2 \cdot \frac{1}{\epsilon})$. □

Thus, the above algorithm runs in polynomial time in $\frac{1}{\epsilon}$. We state the following proposition, whose proof is analogous to the one of Proposition 4.4.

PROPOSITION 4.8. *Allocation function $f^c_{\frac{1-\epsilon}{4}}$ is monotone.*

5. NON–ALLOCATIVELY EFFICIENT MECHANISMS

We are now interested in defining payment rules for the the non–allocatively efficient algorithms presented in Sections 4.2 and 4.3.

By applying the results discussed in [3], we can derive the form of the payments to have DSIC with $f \in \{f_{\frac{1}{4}}, f^c_{\frac{1-\epsilon}{4}}\}$. The payment associated with ad a_i must be of the form:

$$p_{a_i}(v_{a_i}, v_{a_{-i}}) = h_{a_i}(v_{a_{-i}}) + v_{a_i} \cdot CTR_{a_i}(f(v_{a_i}, v_{a_{-i}})) - \int_0^{v_{a_i}} CTR_{a_i}(f(x, v_{a_{-i}})) \cdot dx$$

where h_{a_i} does not depend on v_{a_i}. Assigning $h_{a_i} = 0$ and adopting payments $\overline{p}_{a_i}$ contingent to clicks such that:

$$\overline{p}_{a_i} = \begin{cases} \frac{p_{a_i}(v_{a_i}, v_{a_{-i}})}{CTR_{a_i}(f(v_{a_i}, v_{a_{-i}}))} & \text{if clicked} \\ 0 & \text{otherwise} \end{cases}$$

The payment corresponds to the gray area in Fig. 1, while the above integral corresponds to the white area. The crucial issue here concerns the design of an efficient algorithm

to compute $\int_0^{v_{a_i}} CTR_{a_i}(f(x, v_{a_{-i}})) \cdot dx$. This should be achieved by computing all the possible values of CTR_{a_i} as v_{a_i} varies and the values v_{a_i} in which the CTR_{a_i} changes. The difficulty consists in computing the payments in polynomial time. In this paper we propose two ways for estimating that integral achieving incentive compatibility in expectation.

5.1 IC in expectation, *ex post* WBB, *ex post* IR mechanisms

Adopting the concept of *incentive compatibility in expectation* [3] w.r.t. a random component of the mechanism in place of the concept of DSIC, we can design payments functions with complexity of $O(n^2 \cdot k^3 \cdot \frac{1}{\epsilon})$.

Following the approach presented in [2], we can estimate the term $\int_0^{v_{a_i}} CTR_{a_i}(f(x, v_{a_{-i}})) \cdot dx$ in the payment formula through samples. Call x_{a_i} a value drawn with uniform probability from the support $[0, v_{a_i}]$, it can be shown that:

$$\mathbb{E}_{x_{a_i}}[CTR_{a_i}(f(x_{a_i}, v_{a_{-i}})) \cdot v_{a_i}] = \int_0^{v_{a_i}} CTR_{a_i}(f(x, v_{a_{-i}})) \cdot v_{a_i} \cdot \frac{1}{v_{a_i}} \cdot dx = \int_0^{v_{a_i}} CTR_{a_i}(f(x, v_{a_{-i}})) \cdot dx$$

Therefore, the sample mean is an unbiased estimator of the integral of the previously defined payment formula. In order to compute the payments, it is necessary to sample a value $x_{a_i} \in [0, v_{a_i}]$ for each ad i displayed and then run the allocation function for each couple $(x_{a_i}, v_{a_{-i}})$. Thus, the mechanism requires to run one time the allocation function for determining the allocation of ads and, in the worst case (when all the ads are clicked), k times for computing the payments for each ad allocated.

Adopting a payment strategy $\overline{p}_{a_i}$ contingent to clicks as defined in Section 5, this mechanism satisfies the following properties:

- IR in *ex post*, because u_{a_i} is $\int_0^{x_{a_i}} CTR_{a_i}(f(x, v_{a_{-i}})) \cdot dx$ that is positive, CTR_{a_i} being positive;
- WBB in *ex post*, because $v_{a_i} \cdot CTR_{a_i}(f(v_{a_i}, v_{a_{-i}})) \geq \int_0^{x_{a_i}} CTR_{a_i}(f(x, v_{a_{-i}})) \cdot dx$, f being monotone.

Although the adoption of the above payments allows one to speed up the mechanism, the randomization does not assure a certain profit to the auctioneer especially when the mechanism is repeated few times. Thus, in order to characterize the quality of the above mechanism, it is interesting to provide theoretical bounds in probability over the loss of the auctioneer w.r.t. the exact payments computed as described in the previous section. Let T the number of repetitions, an error bound over the estimator of the payments can be found by applying the Hoeffding's bound [9]. For all the impressed ads a_i, let

$$\Delta_{a_i} = \left| CTR_{a_i}(f(v)) \cdot v_{a_i} - \frac{1}{T} \sum_{j=1}^{T} CTR_{a_i}(f(x_{a_i}, v_{-a_i})) \cdot v_{a_i} - (CTR_{a_i}(f(v)) \cdot v_{a_i} - \int_0^{v_{a_i}} CTR_{a_i}(f(x, v_{-a_i})) \cdot dx) \right|$$

PROPOSITION 5.1. *With probability at least $1-\delta$, it holds*

$$\Delta_{a_i} \leq CTR_{a_i}(f(v)) v_{a_i} \cdot \sqrt{\frac{1}{2T} \cdot \log\left(\frac{2k}{\delta}\right)}$$

for all the impressed ads a_i.

Proof. We have:

$$P(\Delta_{a_i} > t) = P\left(\left| -\frac{1}{T} \sum_{j=1}^{T} CTR_{a_i}(f(x^j, v_{-a_i})) \cdot v_{a_i} + \int_0^{v_{a_i}} CTR_{a_i}(f(x, v_{-a_i})) \cdot dx \right| > t \right) \leq 2e^{-\frac{2T^2t^2}{\sum_{j=1}^{T}(CTR_{a_i}(f(v))v_{a_i}-0)^2}} = 2e^{-\frac{2T^2t^2}{TCTR_{a_i}(f(v))^2 v_{a_i}^2}} = 2e^{-\frac{2Tt^2}{CTR_{a_i}(f(v))^2 v_{a_i}^2}}$$

Thus, $2e^{-\frac{2Tt^2}{CTR_{a_i}(f(v))^2 v_{a_i}^2}} = \frac{\delta}{k}$, $\log e^{-\frac{2Tt^2}{CTR_{a_i}(f(v))^2 v_{a_i}^2}} = \log \frac{\delta}{2k}$, $-\frac{2Tt^2}{CTR_{a_i}(f(v))^2 v_{a_i}^2} = \log \frac{\delta}{2k}$, $t^2 = \frac{CTR_{a_i}(f(v))^2 v_{a_i}^2}{2T} \cdot \log \frac{2k}{\delta}$ and at the end $t = CTR_{a_i}(f(v))v_{a_i} \cdot \sqrt{\frac{1}{2T} \log \frac{2k}{\delta}}$ □

Notice that, as expected, the precision increases as the number T of samples increases and $CTR_{a_i}(f(v))$ decreases, in fact the smaller the $CTR_{a_i}(f(v))$ the smaller the possible error of the estimator and the error decreases with $\sqrt{\frac{1}{2T}}$.

5.2 IC in expectation, *ex post* IR mechanisms, WBB in expectation

In order to reduce further the compute time it is possible to resort to the approach proposed in [4] computing simultaneously the allocation and the payments with the single call to the allocation function f. Thus, the computational complexity of the mechanism is $O(n^2 \cdot k^2 \cdot \frac{1}{\epsilon})$. In addition to incentive compatibility in expectation, the other properties satisfied by this mechanism are:

- IR in *ex post*;
- WBB *in expectation*.

The cost of the computational complexity improvement is the introduction of a randomized component in the allocation function leading to a loss in the total value of the allocation w.r.t. the allocation found by f and that WBB is assured only in expectation w.r.t. the random component.

The random component consists in a modification of the agents' reported values v_{a_i}, each with a (small) probability μ. The modified reported values are then used to compute the allocation and the payments. (Therefore, given an allocation function f, this mechanism returns the same allocation computed by f with a probability of $(1-\mu)^n$, while it can return a different allocation otherwise.) The modification of the reported values is accomplished through a procedure called *canonical self–resampling procedure* described in [4] that generates two samples: $x_{a_i}(v_{a_i}, w_{a_i})$ and $y_{a_i}(v_{a_i}, w_{a_i})$, where w_{a_i} is the random seed.

Algorithm 3 computeAllocationAndPayments$(v_{a_1}, \ldots, v_{a_n})$

1: **for all** $i \in N$ **do**
2: $x_i = canonicalSRP(v_{a_i})$ [4]
3: find the optimal allocation with values x: $f^c_{\frac{1-\epsilon}{4}}$
4: **for all** $i \in N$ **do**
5: compute payments: $p_{a_i} = CTR_{a_i}(f(x)) \cdot v_{a_i} - estimatedInt_{a_i}$, where $estimatedInt_{a_i} = \begin{cases} \frac{CTR_{a_i}(f(x)) \cdot v_{a_i}}{\mu} & \text{if } y_{a_i} < v_{a_i}, \\ 0 & \text{otherwise.} \end{cases}$

Algorithm 3 perturbs the bid through the canonical self–resampling procedure (Step 2). Given the perturbed bids

(x), it finds the new optimal allocation (on average it is a least a $(1-\frac{\mu}{2-\mu})\cdot\frac{1-\epsilon}{4}$ approximation of the true optimal one) and then compute the payments as described at Step 5. The payment is negative (when $y_{a_i} < v_{a_i}$) or equal to $CTR_{a_i}(f(x))v_{a_i}$ otherwise, thus $p_{a_i} \leq CTR_{a_i}(f(x))v_{a_i}$. It is possible to compute the contingent payments $\overline{p}_{a_i}$ dividing p_{a_i} by $CTR_{a_i}(x)$ with the x computed when ad a_i is clicked, in this way we obtain a mechanism *ex post* IR. The payment formula shows that with high probability each agent pays the mechanism more than what it would do with the payment functions described in Section 5.1, but sometimes, it receives a large amount of money from the auctioneer. This amount is $v_{a_i}\cdot CTR_{a_i}(f(x))\cdot(\frac{1}{\mu}-1)$. Hence, there are positive transfers from the auctioneer to the advertisers and there is no guarantee that the auctioneer receives more then what he pays. This makes the mechanism WBB only in expectation.

Also in this case, we derive an error bound over the estimator of the payment applying the Hoeffding's bound.

PROPOSITION 5.2. *With probability at least* $1-\delta$*, it holds*

$$\left|\frac{1}{T}\sum_{j=1}^{T}\left(CTR_{a_i}(x^j)\cdot v_{a_i} - esitmatedInt_{a_i}(x^j,y^j)\right) - \mathbb{E}[p_{a_i}]\right| \leq \frac{v_{a_i}}{\mu}\cdot\sqrt{\frac{1}{2T}\log\frac{2n}{\delta}}$$

for all the ads.

Proof. We have:

$$P\left(\left|\frac{1}{T}\sum_{j=1}^{T}(CTR_{a_i}(f(x^j))\cdot v_{a_i} - esitmatedInt_{a_i}(x^j,y^j)) - \mathbb{E}[p_{a_i}]\right| > t\right) = 2e^{-\frac{2T^2t^2}{\sum_{j=1}^{T}(v_{a_i}-v_{a_i}+\frac{v_{a_i}}{\mu})^2}} = 2e^{-\frac{2T^2t^2}{T(\frac{v_{a_i}}{\mu})^2}}$$

Thus, $2e^{-\frac{2Tt^2}{(\frac{v_{a_i}}{\mu})^2}} = \frac{\delta}{n}$, $\log e^{-\frac{2Tt^2}{(\frac{v_{a_i}}{\mu})^2}} = \log\frac{\delta}{2n}$, $-\frac{2Tt^2}{(\frac{v_{a_i}}{\mu})^2} = \log\frac{\delta}{2n}$, $t^2 = (\frac{v_{a_i}}{\mu})^2\frac{1}{2T}\log\frac{2n}{\delta}$, and $t = \frac{v_{a_i}}{\mu}\sqrt{\frac{1}{2T}\log\frac{2n}{\delta}}$. □

This mechanism presents a larger variance in payments than that described in Section 5.1. Here, the bound depends also on $\frac{1}{\mu}$ and therefore, keeping μ close to zero to have a fine approximation of the best allocation returned by f, $\frac{1}{\mu}$ can be very large.

6. EXPERIMENTAL ANALYSIS

Experimental setting.The experimental evaluation is based on the *Yahoo! Webscope A3* dataset. For every day, advertiser, keyword, and position Yahoo! Webscope A3 specifies, over a period of 4 months, the average bid value, the number of displays without click and the number of clicks, but it does not specify the set of ads of each allocation. Thus, we did not use directly the dataset, but we exploited it to develop a generator of realistic synthetic instances. We considered the 100 keywords with the highest numbers of impressions, and for each of these keywords we generated a separate bid and quality distribution. Each bid distribution consists of a truncated Gaussian distribution, where the mean and standard deviation are taken from the dataset, the lower bound corresponds to the minimum bid value in the dataset, and the upper bound to the highest one. Furthermore, we used a beta distribution from which to sample the quality. We assumed $\lambda_{s_1} = 1$, thus the quality corresponds to the click probability when the ad is displayed in the first position, and we used the data from the first position to derive the parameters of the beta distribution. We estimated the prominence λ_{s_j} by computing the average of $\frac{CTR_{a_i}(\theta)|\theta(a_i)=s_j}{q_{a_i}}$ over all the ads and all the auctions.[2] Instead, each continuation probability is generated as follows: we have considered two different scenarios, the first in which we suppose most of the ads having a high continuation probability, i.e. c_{a_i} is uniformly drawn in $[0.7, 1.0]$ with probability 0.9 and from $[0.0, 0.7]$ with probability 0.1, while in the second we suppose that ads have a c_{a_i} uniformly distributed in $[0.0, 1.0]$. We varied k and n as: $k \in \{2,\ldots,10\}$ with a step of 2 and $n \in \{5,\ldots,200\}$ with a step of 5. For each pair k, n we generated 20 instances. Finally, we implemented our algorithms in C language and executed them with Intel 2.20GHz and Linux kernel 2.6.32.

Allocation efficiency. We experimentally compared the efficiency of the allocations returned by the $f_{\frac{1}{4}}$ and $f^c_{\frac{1-\epsilon}{4}}$ with $\epsilon \in \{0.001, 0.1, 0.3, 0.5, 0.7, 0.9\}$ w.r.t. the efficient allocation returned by f_E. Each plot in Fig. 3 reports how the average efficiency (i.e., $\frac{f_{\frac{1}{4}}}{f_E}$ and $\frac{f^c_{\frac{1-\epsilon}{4}}}{f_E}$) varies as the number of ads varies, while the plots differentiate for the number of available slots; c_{a_i} are drawn uniformly from $[0.0, 1.0]$.

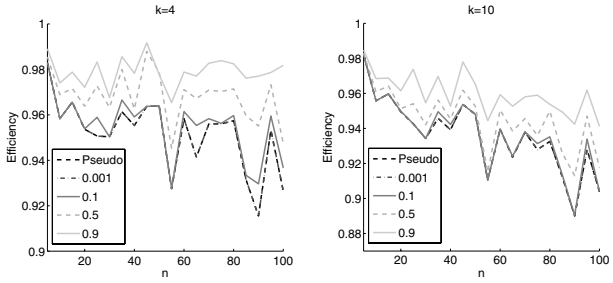

Figure 2: Average efficiency ratio between $f^c_{\frac{1-\epsilon}{4}}$ and f_E with different values of ϵ and n for $k \in \{4, 10\}$ when c_{a_i} are drawn uniformly from $[0.0, 1.0]$.

Fig. 2 shows that, surprisingly, the efficiency increases increasing ϵ. A motivation is that increasing ϵ we are relaxing the constraint $\sum_{i=1}^{l}\log_2\frac{1}{c_{a_i}} \leq 1$, thus we allow the allocation with more ads that can reach the last ad with a continuation probability probability lower than $\frac{1}{2}$. The approximation is, on average, always higher than 0.88 even with $k = 10$ for $\epsilon \in \{0.001, 0.1, 0.5, 0.9\}$. The results obtained with the first scenario for the generation of c_{a_i} are consistent with the above ones. Thus, the efficient allocation can be satisfactorily approximated in realistic scenarios.

Then, we have evaluated how the randomization μ used in the payments defined in Section 5.2 affects the efficiency of the allocation. We observed that, with both c_{a_i} generation scenarios, on average with $\mu \leq 0.1$ the loss of efficiency is small (<5%) while it is $\leq 15\%$ until $\mu \leq 0.7$.

Compute time. We experimentally evaluated the compute time to find an allocation with $f_E, f_{\frac{1}{4}}, f^c_{\frac{1-\epsilon}{4}}$ with the values of ϵ used above. As expected, it rises as k and n rise. Each plot in Fig. 3 reports how the average compute time varies as the number of ads varies, while the plots differentiate for the number of available slots.

The plots in Fig. 4 report the same data, but with the reverse approach, and they focus more in the exponential

[2] $\lambda_{s_j} \in \{1, 0.714, 0.556, 0.525, 0.494, 0.470, 0.444, 0.441, 0.432, 0.427\}$.

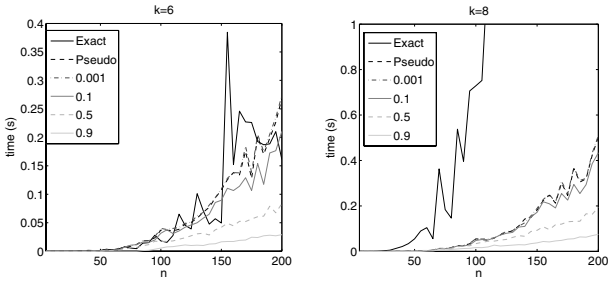

Figure 3: Average compute time between $f^c_{\frac{1-\epsilon}{4}}$ and f_E with different values of ϵ and n for $k \in \{2, 4, 6, 8\}$.

behavior of f_E. With $k = 10$ and $n = 100$ f_E can require even more than 30 s. The approximation algorithms are so faster than the exact one that cannot be seen in the plots.

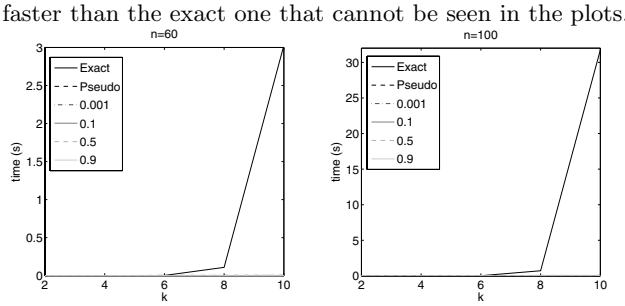

Figure 4: Average compute time between $f^c_{\frac{1-\epsilon}{4}}$ and f_E with different values of ϵ and k for $k \in \{60, 100\}$.

In the pictures it is difficult to distinguish $f^c_{\frac{1-\epsilon}{4}}$ with $\epsilon = 0.001$ from $f_{\frac{1}{4}}$ curve because their behavior is almost the same. It can be observed that with $k = 6$ and $n = 100$ — the allocations are $O(10^{12})$ — the efficient allocation can be computed in real time, requiring less than 0.01 s on average. With $k > 7$ approximation algorithms are necessary. It can be observed that, even with $n = 200$ and $k = 10$, compute time is < 0.4 s when $\epsilon \geq 0.1$ and < 0.1 s when $\epsilon \geq 0.9$. Thus with $k \leq 7$ or a small number of ads the optimal strategy is to use f_E that is fast enough and optimal, while in the other situation use $f^c_{\frac{1-\epsilon}{4}}$ with increasing value of ϵ as n increases.

In order to complete the mechanism, we need to choose a payment rule when $f^c_{\frac{1-\epsilon}{4}}$ is adopted. On the basis of our results, the payments described in Section 5.2 are the fastest and the efficiency loss of the allocation, even with a large μ, is moderately low. However, the very high variance makes these payments non–attractive. Hence, the best payments are those described in Section 5.1.

7. CONCLUSIONS AND FUTURE WORKS

In this paper we analyzed the problem to design incentive compatible mechanisms in sponsored search auctions, where the user behavior is modeled also taking into account ad externalities through the cascade model. To be best of our knowledge, we provided the first algorithm for finding optimal allocations. Applying it together with the VCG payment rule we obtained an incentive compatible mechanism with exponential complexity. To make the problem tractable we showed that it is possible to design a *monotone* $\frac{1-\epsilon}{4}$–approximation allocation algorithms. For this algorithm we designed two payment rules that allow to build two incentive compatible mechanisms in expectation. Finally, we provided an experimental evaluation of the presented algorithms with *Yahoo! Webscope A3* dataset, showing that the exact algorithm can be used in real time within a given parameter ranges. When the number of slots or ads becomes too large approximation algorithms are necessary. Their adoption can be performed in real time requiring a very short compute time and finding very efficient allocations.

In future, we will study learning techniques to estimate the cascade model parameters, i.e. q_{a_i}, c_{a_i}, and λ_{s_i}, resorting to the current state of the art ([4] and [8]). Another interesting task is the determination of the complexity class of the problem of finding the optimal solution for the cascade model with both ad and position externalities, being currently supposed to be $\mathcal{NP}$–hard, but no proof being known.

8. REFERENCES

[1] G. Aggarwal, J. Feldman, S. Muthukrishnan, and M. Pál. Sponsored search auctions with markovian users. In *WINE*, pages 621–628, 2008.

[2] A. Archer, C. Papadimitriou, K. Talwar, and É Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *SODA*, pages 205–214, 2003.

[3] A. Archer and É Tardos. Truthful mechanisms for one–parameter agents. In *FOCS*, pages 482–, 2001.

[4] M. Babaioff, R. D. Kleinberg, and A. Slivkins. Truthful mechanisms with implicit payment computation. In *ACM EC*, pages 43–52, 2010.

[5] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position–bias models. In *WSDM*, pages 87–94, 2008.

[6] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second–price auction: Selling billions of dollars worth of keywords. *AM ECON REV*, 97(1):242–259, 2007.

[7] D. Fotakis, P. Krysta, and O. Telelis. Externalities among advertisers in sponsored search. In *SAGT*, pages 105–116, 2011.

[8] N. Gatti, A. Lazaric, and F. Trovò. A truthful learning mechanism for contextual multi–slot sponsored search auctions with externalities. In *ACM EC*, pages 605–622, 2012.

[9] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J AM STAT ASSOC*, 58(301):13–30, 1963.

[10] Interactive Advertising Bureau. IAB internet advertising revenue report. 2011 full year results. Technical report, 2012.

[11] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM T INFORM SYST*, 25(2), 2007.

[12] D. Kempe and M. Mahdian. A cascade model for externalities in sponsored search. In *WINE*, pages 585–596, 2008.

[13] R. Paes Leme and É. Tardos. Pure and Bayes–Nash price of anarchy for generalized second price auction. In *FOCS*, pages 735–744, 2010.

[14] Y. Narahari, D. Garg, R. Narayanam, and H. Prakash. *Game Theoretic Problems in Network Economics and Mechanism Design Solutions*. Springer, February 2009.

[15] V. V. Vazirani. Knapsack. In *Approximation Algorithms*, pages 68–73. Springer, 2001.