Click Fraud Resistant Methods for Learning Click-Through Rates

Nicole Immorlica, Kamal Jain, Mohammad Mahdian, and Kunal Talwar

Microsoft Research, Redmond, WA, USA,
{nickle,kamalj,mahdian,kunal}@microsoft.com

Abstract. In pay-per-click online advertising systems like Google, Overture, or MSN, advertisers are charged for their ads *only* when a user clicks on the ad. While these systems have many advantages over other methods of selling online ads, they suffer from one major drawback. They are highly susceptible to a particular style of fraudulent attack called *click fraud*. Click fraud happens when an advertiser or service provider generates clicks on an ad with the sole intent of increasing the payment of the advertiser. Leaders in the pay-per-click marketplace have identified click fraud as the most significant threat to their business model. We demonstrate that a particular class of learning algorithms, called *click-based algorithms*, are resistant to click fraud in some sense. We focus on a simple situation in which there is just one ad slot, and show that fraudulent clicks can not increase the expected payment per impression by more than $o(1)$ in a click-based algorithm. Conversely, we show that other common learning algorithms are vulnerable to fraudulent attacks.

1 Introduction

The Internet is probably the most important technological creation of our times. It provides many immensely useful services to the masses for free, including such essentials as web portals, web email and web search. These services are expensive to maintain and depend upon advertisement revenue to remain free. Many services such as Google, Overture, and certain components of MSN, generate advertisement revenue by selling clicks. In these *pay-per-click* systems, an advertiser is charged only when a user clicks on his ad.

A scenario of particular concern for service providers and advertisers in pay-per-click markets is *click fraud* – the practice of gaming the system by creating fraudulent clicks, usually with the intent of increasing the payment of the advertiser. As each click can cost on the order of $1, it does not take many fraudulent clicks to generate a large bill. Just a million fraudulent clicks, generated perhaps by a simple script, can cost the advertiser $1,000,000, easily exhausting his budget. Fraudulent behavior threatens the very existence of the pay-per-click advertising market and has consequently become a subject of great concern [5, 7, 8]. Recently, Google CFO George Reyes said, in regards to the click fraud problem, that "I think something has to be done about this really, really quickly, because I think, potentially, it threatens our business model." [8].

A variety of proposals for reducing click fraud have surfaced. Most service providers currently approach the problem of click fraud by attempting to automatically recognize fraudulent clicks and discount them. Fraudulent clicks are recognized by machine learning algorithms, which use information regarding the navigational behavior of users to try and distinguish between human and robot-generated clicks. Such techniques require large datasets to train the learning methods, have high classification error, and are at the mercy of the "wisdom" of the scammers. Recent tricks, like using cheap labor in India to generate these fraudulent clicks [9], make it virtually impossible to use these machine learning algorithms.

Another line of proposals attempts to reduce click fraud by removing the incentives for it. Each display of an ad is called an *impression*. Goodman [2] proposed selling advertisers a particular *percentage* of all impressions rather than user clicks. Similar proposals have suggested selling impressions. For a click-through-rates of 1%, the expected price per impression in the scenario mentioned above is just one cent. Thus, to force a payment of $1,000,000 upon the advertiser, 100,000,000 fraudulent impressions must be generated versus just 1,000,000 fraudulent clicks in the pay-per-click system. When such large quantities of fraud are required to create the desired effect, it ceases to be profitable to the scammer.

Although percentage and impression based proposals effectively eliminate fraud, they suffer from three major drawbacks. First, the developed industry standard sells clicks, and any major departure from this model risks a negative backlash in the marketplace. Second, by selling clicks, the service provider subsumes some of the risk due to natural fluctuations in the marketplace (differences between day and night or week and weekend, for example). Third, by requesting a bid per click, the service provider lessons the difficulty of the strategic calculation for the advertiser. Namely, the advertiser only needs to estimate the worth of a click, an arguably easier task than estimating the worth of an impression.

In this paper, we attempt to eliminate the incentives for click fraud in a system that sells clicks. We focus on a common pay-per-click system, generally believed to be used by Google [3] among others, which has been shown empirically to have higher revenue [1, 4] than other pay-per-click systems like that of Overture [6]. This system is based on estimates of the *click-through rate* (CTR) of an ad. The CTR is defined as the likelihood, or probability, that an impression of an ad generates a click. In this system, each advertiser submits a bid which is the maximum amount the advertiser is willing to pay per click of the ad. The advertisers are then ranked based on the product of their bids and respective estimated CTRs of their ads. This product can be interpreted as an expected bid per impression. The ad space is allocated in the order induced by this ranking. Advertisers are charged *only* if they receive a click, and they are charged an amount inversely proportional to their CTR.

In pay-per-click systems, when a fraudulent click happens, an advertiser has to pay for it, resulting in a short term loss to the advertiser whose ad is being clicked fraudulently. However, in the system described above, there is a long term benefit too. Namely, a fraudulent click will be interpreted as an increased

likelihood of a future click and so result in an increase in the estimate of the CTR. As the payment is inversely proportional to the CTR, this results in a reduction in the payment. If the short term loss and the long term benefit exactly cancel each other, then there will be less incentive to generate fraudulent clicks; in fact, a fraudulent click or impression will only cost the advertiser as much as a fraudulent impression in a pay-per-impression scheme. Whether this happens depends significantly on how the system estimates the CTRs. There are a variety of sensible algorithms for this task. Some options include taking the fraction of all impressions so far that generated a click, or the fraction of impressions in the last hour that generated a click, or the fraction of the last hundred impressions that generated a click, or the inverse of the number of impressions after the most recent click, and so on.

We prove that a particular class of learning algorithms, called *click-based* algorithms, have the property that the short term loss and long term benefit in fact cancel. Click-based algorithms are a class of algorithms whose estimates are based upon the number of impressions between clicks. To compute the current estimate, a click-based algorithm computes a weight for each impression based solely on the number of clicks after it and then takes the weighted average. An example of an algorithm in this class is the one which outputs an estimate equal to the reciprocal of the number of impressions before the most recent click. We show that click-based algorithms satisfying additional technical assumptions are *fraud-resistant* in the sense that a devious user can not change the expected payment of the advertiser per impression (see Section 3 for a formal definition). We provide an example that a traditional method for estimating CTR (that is, taking the average over a fixed number of recent impressions) is not fraud-resistant.

The structure of this paper is as follows. In Section 2, we describe the setting. In Section 3, we define a very general class of algorithms for learning the CTR of an ad, called CTR learning algorithms. In Section 4, we define a special class of these algorithms, called click-based algorithms, and prove that they are fraud-resistent. In Section 5, we give examples showing that other common algorithms for learning the CTR are not fraud-resistent.

2 Setting

We consider a simple setting in which a service provider wishes to sell space for a single ad on a web page. There are a number of advertisers, each of whom wishes to display their ad on the web page. The service provider sells the ad space according to the pay-per-click model and through an auction: the advertiser whose ad is displayed is charged only when a user clicks on his ad. Each advertiser i submits a bid b_i indicating the maximum amount he is willing to pay the service provider when a user clicks on his ad. The allocation and price is computed using the mechanism described below.

For each ad, the service provider estimates the probability that the ad receives a click from the user requesting the page, if it is displayed. This probability is

called the *click-through-rate (CTR)* of the ad. Each bid b_i is multiplied by the estimate λ_i of the CTR of the ad. The product $\lambda_i b_i$ thus represents the expected willingness-to-pay of advertiser i per impression. The slot is awarded to the advertiser i^* with the highest value of $\lambda_i b_i$. If the user indeed clicks on the ad, then the winning advertiser is charged a price equal to the second highest $\lambda_i b_i$ divided by his (that is, the winner's) estimated CTR (that is, λ_{i^*}). Thus, if we label advertisers such that $\lambda_i b_i > \lambda_{i+1} b_{i+1}$, then the slot is awarded to advertiser 1 and, upon a click, he is charged a price $\lambda_2 b_2 / \lambda_1$. In this paper, we study the mechanism over a period of time during which the same advertiser wins the auction, and the value of $\lambda_2 b_2$ does not change. If the advertisers do not change their bids too frequently and $\lambda_1 b_1$ and $\lambda_2 b_2$ are not too close to each other, it is natural to expect this to happen most of the time. We will henceforth focus on the winner of the auction, defining $p := \lambda_2 b_2$ and $\lambda := \lambda_1$.

3 CTR Learning Algorithms

Of course, we have left unspecified the method by which the algorithm learns the CTRs. The subject of this work is to study different algorithms for computing the CTR of an advertiser. There are a variety of different algorithms one could imagine for learning the CTR of an ad. Some simple examples, described below, include averaging over time, impressions, or clicks, as well as exponential discounting.

- **Average over fixed time window:** For a parameter T, let x be the number of clicks received during the last T time units and y be the number of impressions during the last T time units. Then $\lambda = x/y$.
- **Average over fixed impression window:** For a parameter y, let x be the number of clicks received during the last y impressions. Then $\lambda = x/y$.
- **Average over fixed click window:** For a parameter x, let y be the number of impressions since the x'th last click. Then $\lambda = x/y$.
- **Exponential discounting:** For a parameter α, let $e^{-\alpha i}$ be a discounting factor used to weight the i'th most recent impression. Take a weighted average over all impressions, that is, $\sum_i x_i e^{-\alpha i} / \sum_i e^{-\alpha i}$ where x_i is an indicator variable that the i'th impression resulted in a click.

These algorithms are all part of a general class defined below. The algorithm estimates the CTR of the ad for the current impression as follows: Label the previous impressions, starting with the most recent, by $1, 2, \ldots$. Let t_i be the amount of time that elapsed between impression i and impression 1, and c_i be the number of impressions that received clicks between impression i and impression 1 (impressions 1 included). The learning algorithms we are interested in are defined by a constant γ and a function $\delta(t_i, i, c_i)$ which is decreasing in all three parameters. This function can be thought of as a discounting parameter, allowing the learning algorithm to emphasize recent history over more distant history. Let x_i be an indicator variable for the event that the i'th impression

resulted in a click. The learning algorithm then computes

$$\lambda = \frac{\sum_{i=1}^{\infty} x_i \delta(t_i, i, c_i) + \gamma}{\sum_{i=1}^{\infty} \delta(t_i, i, c_i) + \gamma}.$$

The constant γ is often a small constant that is used to guarantee that the estimated click-through-rate is strictly positive and finite. Notice that in the above expression, the summation is for every i from 1 to ∞. This is ambiguious, since the advertiser has not been always present in the system. To remove this ambiguity, the algorithm assumes a *default* infinite history for every advertiser that enters the system. This default sequence could be a sequence of impressions all leading to clicks, indicating that the newly arrived advertiser is initialized with a CTR equal to one, or (as it is often the case in practice) it could be a sequence indicating a system-wide default initial CTR for new advertisers. For most common learning algorithms, the discount factor becomes zero or very small for far distant history, and hence the choice of the default sequence only affects the estimate of the CTR at the arrival of a new advertiser.

Note that all three learning methods discussed above are included in this class (for $\gamma = 0$).

- **Average over fixed time window:** The function $\delta(t_i, i, c_i)$ is 1 if $t_i \leq T$ and 0 otherwise.
- **Average over fixed impression window:** The function $\delta(t_i, i, c_i)$ is 1 if $i \leq y$ and 0 otherwise.
- **Average over fixed click window:** The function $\delta(t_i, i, c_i)$ is 1 if $c_i \leq x$ and 0 otherwise.
- **Exponential discounting:** The function $\delta(t_i, i, c_i)$ is $e^{-\alpha i}$.

4 Fraud Resistance

For each of the methods listed in the previous section, for an appropriate setting of parameters (e.g., large enough y in the second method), on a random sequence generated from a constant CTR the estimate computed by the algorithm gets arbitrarily close to the true CTR, and so it is not a priori apparent which method we might prefer. Furthermore, when the learning algorithm computes the true CTR, the expected behavior of the system is essentially equivalent to a pay-per-impression system, with substantially reduced incentives for fraud. This might lead to the conclusion that all of the above algorithms are equally resistant to click fraud. However, this conclusion is wrong, as the scammer can sometimes create fluctuations in the CTR, thereby taking advantage of the failure of the algorithm to react quickly to the change in the CTR to harm the advertiser.

In this section, we introduce a notion of fraud resistance for CTR learning algorithms, and prove that a class of algorithms are fraud-resistant. The definition of fraud resistance is motivated by the way various notions of security are defined in cryptography: we compare the expected amount the advertiser has to pay in two scenarios, one based on a random sequence generated from a constant CTR without any fraud, and the other with an adversary who can change

a fraction of the outcomes (click vs. no-click) on a similar random sequence. Any scenario can be described by a time-stamped sequence of the outcomes of impressions (i.e., click or no-click). More precisely, if we denote a click by 1 and a no-click by 0, the scenario can be described by a doubly infinite sequence $\mathbf{s}$ of zeros and ones, and a doubly infinite increasing sequence $\mathbf{t}$ of real numbers indicating the time stamps (the latter sequence is irrelevant if the learning algorithm is time-independent, which will be the case for the algorithms we consider in this section). The pair $(\mathbf{s}, \mathbf{t})$ indicates a scenario where the i'th impression (i can be any integer, positive or negative) occurs at time t_i and results in a click if and only if $s_i = 1$.

Definition 1. *Let ϵ be a constant between zero and one, and $(\mathbf{s}, \mathbf{t})$ be a scenario generated at random as follows: the outcome of the ith impression, s_i, is 1 with an arbitrary fixed probability λ and 0 otherwise, and the time difference $t_i - t_{i-1}$ between two consecutive impressions is drawn from a Poisson distribution with an arbitrary fixed mean. For a value of n, let $(\mathbf{s}', \mathbf{t}')$ be a history obtained from $(\mathbf{s}, \mathbf{t})$ by letting an adversary insert at most ϵn impressions after the impression indexed 0 in $(\mathbf{s}, \mathbf{t})$. The history $(\mathbf{s}', \mathbf{t}')$ is indexed in such a way that impression 0 refers to the same impression in $(\mathbf{s}, \mathbf{t})$ and $(\mathbf{s}', \mathbf{t}')$. We say that a CTR learning algorithm is* ϵ-fraud resistant *if for every adversary, the expected average payment of the advertiser per impression during the impressions indexed $1, \ldots, n$ in scenario $(\mathbf{s}', \mathbf{t}')$ is bounded by that of scenario $(\mathbf{s}, \mathbf{t})$, plus an additional term that tends to zero as n tends to infinity (holding everything else constant). More precisely, if q_j (q'_j, respectively) denotes the payment of the advertiser for the jth impression in scenario $(\mathbf{s}, \mathbf{t})$ ($(\mathbf{s}', \mathbf{t}')$, respectively), then the algorithm is ϵ-fraud resistant if for every adversary,*

$$\mathrm{E}[\frac{1}{n}\sum_{j=1}^{n} q'_j] \leq \mathrm{E}[\frac{1}{n}\sum_{j=1}^{n} q_j] + o(1).$$

Intuitively, in a fraud-resistant algorithm, a fraudulent click or impression only costs the advertiser as much as a fraudulent impression in a pay-per-impression scheme.

Some details are intentionally left ambiguous in the above definition. In particular, we have not specified how much knowledge the adversary has. In practice, an adversary probably can gain knowledge about some statistics of the history, but not the complete history. However, our positive result in this section holds even for an all powerful adversary that knows the whole sequence (even the future) in advance. We prove that even for such an adversary, there are simple learning algorithms that are fraud-resistant. Our negative result (presented in the next section), shows that many learning algorithms are not fraud-resistant even if the adversary only knows about the learning algorithm and the frequency of impressions in the scenario. Therefore, our results are quite robust in this respect.

Another point that is worth mentioning is that the assumption that the true click-through rate λ is a constant in the above definition is merely a simplifying

assumption. In fact, our results hold (with the same proof) even if the parameter λ changes over time, as long as the value of λ at every point is at least a positive constant (i.e., does not get arbitrarily close to zero). Also, the choice of the distribution for the time stamps in the definition was arbitrary, as our positive result only concerns CTR learning algorithms that are time-independent, and our negative result in the next section can be adapted to any case where the time stamps come from an arbitrary known distribution.

In this section, we show that CTR learning algorithms for which the discounting factor, δ, depends only on the number of impressions in the history which resulted in clicks, that is the parameter c_i defined above (and not on i and t_i), are fraud-resistant. We call such algorithms *click-based* algorithms.

Definition 2. *A CTR learning algorithm is click-based if $\delta(t_i, i, c_i) = \delta(c_i)$ for some decreasing function $\delta(.)$.*

Of the schemes listed in the previous section, it is easy to see that only averaging over clicks is click-based. Intuitively, a click-based algorithm estimates the CTR by estimating the *Expected Click-Wait* (ECW), the number of impression it takes to receive a click.

The following theorem shows that click-based algorithms are fraud-resistant.

Theorem 1. *Consider a click-based CTR learning algorithm A given by a discounting function $\delta(.)$ and $\gamma = 0$. Assume that $\sum_{i=1}^{\infty} i\delta(i)$ is bounded. Then for every $\epsilon \leq 1$, the algorithm A is ϵ-fraud-resistant.*

Proof. The proof is based on a simple "charging" argument. We distribute the payment for each click over the impressions preceding it, and then bound the expected total charge to any single impression due to the clicks after it.

We begin by introducing some notations. For any scenario $(\mathbf{s}, \mathbf{t})$ and index i, let $S_{i,j}$ denote the set of impressions between the j'th and the $(j-1)$'st most recent click before impression i (including click j but not click $j-1$) and $n_{i,j} = |S_{i,j}|$. Then the estimated CTR at i can be written as

$$\frac{\sum_{j=1}^{\infty} \delta(j)}{\sum_{j=1}^{\infty} n_{i,j}\delta(j)}. \tag{1}$$

and hence the payment at i if impression i receives a click is

$$p \times \frac{\sum_{j=1}^{\infty} n_{i,j}\delta(j)}{\sum_{j=1}^{\infty} \delta(j)}.$$

We now introduce a scheme to charge this payment to the preceding impressions (for both with-fraud and without-fraud scenarios). Fix an impression i' for $i' < i$ and let j be the number of clicks between i' and i (including i' if it is a click). If impression i leads to a click, we charge i' an amount equal to

$$p \times \frac{\delta(j)}{\sum_{j=1}^{\infty} \delta(j)}. \tag{2}$$

for this click. Summing over all impressions preceding click i, we see that the total payment charged is equal to the payment for the click at impression i. The crux of the argument in the remainder of the proof is to show that in both scenarios, with or without fraud, the average total payment charged to an impression i in the interval $[1, n]$ by clicks occurring after i is $p \pm o(1)$. We start by proving an upper bound on the total amount charged to the impressions.

We first focus on bounding the total amount charged to impressions before impression 0. The impressions in the set $S_{0,j}$ are charged by the i'th click after impression 0 a total of

$$p \times \frac{n_{0,j}\delta(i+j)}{\sum_{k=1}^{\infty}\delta(k)}.$$

Summing over all clicks between impression 0 and impression n (inclusive), we see that the total charge to the impressions in $S_{0,j}$ is at most

$$p \times \frac{\sum_{i=1}^{\infty} n_{0,j}\delta(i+j)}{\sum_{k=1}^{\infty}\delta(k)}.$$

Summing over all sets $S_{0,j}$, we see that the total charge to impressions before 0 is at most

$$p \times \frac{\sum_{j=1}^{\infty}\sum_{i=1}^{\infty} n_{0,j}\delta(i+j)}{\sum_{k=1}^{\infty}\delta(k)}.$$

Since the denominator $\sum_{k=1}^{\infty}\delta(k)$ is a positive constant, we only need to bound the expected value of the numerator $\sum_{j=1}^{\infty}\sum_{i=1}^{\infty} n_{0,j}\delta(i+j)$.

Since in both with-fraud and without-fraud scenarios the probability of click for each impression before impression 0 is λ, the expected value of $n_{0,j}$ in both scenarios is $1/\lambda$. Therefore, the expectation of the total amount charged to impressions before 0 in both scenarios is at most

$$\frac{p}{\lambda} \times \frac{\sum_{j=1}^{\infty}\sum_{i=1}^{\infty}\delta(i+j)}{\sum_{k=1}^{\infty}\delta(k)} = \frac{p}{\lambda} \times \frac{\sum_{k=1}^{\infty} k\delta(k)}{\sum_{k=1}^{\infty}\delta(k)},$$

which is bounded by a constant (i.e., independent of n) since $\sum_{k=1}^{\infty} k\delta(k)$ is finite.

We now bound payments charged to impressions after impression 0. For a fixed impression i with $0 \le i \le n$, the payment charged to i by the j'th click after i is given by Equation 2. Summing over all j, we see that the total payment charged to i is at most

$$p \times \frac{\sum_{j=1}^{\infty}\delta(j)}{\sum_{j=1}^{\infty}\delta(j)} = p. \tag{3}$$

By the above equation along with the fact that the expected total charge to impressions before impression 0 is bounded, we see that the expected total charge of all impressions is at most $np+O(1)$, and therefore the expected average payment per impression (in both with-fraud and without-fraud scenarios) is at most $p + o(1)$.

We now show that in the scenario $(\mathbf{s}, \mathbf{t})$ (i.e., without fraud), the expected average payment per impression is at least $p - o(1)$. Let k be the number of clicks in the interval $[1, n]$ and consider an impression in $S_{n,j}$. Then, by Equation 2, this impression is charged an amount equal to

$$p \times \frac{\sum_{i=1}^{j-1} \delta(i)}{\sum_{i=1}^{\infty} \delta(i)} = p - \frac{p \sum_{i=j}^{\infty} \delta(i)}{\sum_{i=1}^{\infty} \delta(i)}.$$

Therefore, the total amount charged to the impressions in the interval $[1, n]$ is at least

$$np - \frac{p \sum_{j=1}^{k+1} n_{n,j} \sum_{i=j}^{\infty} \delta(i)}{\sum_{i=1}^{\infty} \delta(i)}.$$

As in the previous case, the expected value of $n_{n,j}$ in the scenario without any fraud is precisely $1/\lambda$. Therefore, the expectation of the total charge to impressions in $[1, n]$ is at least

$$np - \frac{p \sum_{j=1}^{k+1} \sum_{i=j}^{\infty} \delta(i)}{\lambda \sum_{i=1}^{\infty} \delta(i)} \geq np - \frac{p}{\lambda} \times \frac{\sum_{i=1}^{\infty} i\delta(i)}{\sum_{i=1}^{\infty} \delta(i)}.$$

Therefore, since $\sum_{i=1}^{\infty} i\delta(i)$ is bounded, the expected average payment per impression in the scenario without fraud is at least $p - o(1)$. This shows that the difference between the expected average payment per impression in the two scenarios is at most $o(1)$, and hence the algorithm is ϵ-fraud resistant.

5 Non-Click-Based Algorithms

In this section, we give an example that shows that in many simple non-click-based algorithms (such as averaging over fixed time window or impression window presented in Section 3), an adversary can use a simple strategy to increase the average payment of the advertiser per impression. We present the example for the learning algorithm that takes the average over a fixed impression window. It is easy to see that a similar example exists for averaging over a fixed time window.

Consider a history defined by setting the outcome of each impression to click with probability λ for a fixed λ. Denote this sequence by $\mathbf{s}$. We consider the algorithm that estimates the CTR by the number of click-throughs during the past l impressions plus a small constant γ divided by $l + \gamma$, for a fixed l. If l is large enough and γ is small but positive, the estimate provided by the algorithm is often very close to λ, and therefore the average payment per impression on any interval of length n is arbitrarily close to p. In the following proposition, we show that an adversary can increase the average payment by a non-negligible amount.

Proposition 1. *In the scenario defined above, there is an adversary that can increase the average payment per impression over any interval of length n, for*

any large enough n, by inserting ϵn fradulent impressions and clicking on some of them.

Proof Sketch: Consider the following adversary: The adversary inserts ϵn fradulent impressions distributed uniformly in the interval starting at the impression indexed 1 and ending at the impression indexed $(1-\epsilon)n$ (with an outcome described below), so that in the scenario with fraud, each of the first n impressions after impression 0 is fradulent with probability ϵ. Divide the set of impressions after impression 0 into a set of intervals $I_1, I_2, \ldots$, where each interval I_j contains l impressions (real or fradulent). In other words, I_1 is the set of the first l impressions after impression 0, I_2 is the set of the next l impressions, etc. The adversary sets the outcome of all fradulent impression in I_j for j odd to click and for j even to no-click. This means that the "true" CTR during I_j is $(1-\epsilon)\lambda+\epsilon$ for odd j and $(1-\epsilon)\lambda$ for even j. The algorithm estimates the CTR for the r'th impression of the interval I_j by dividing the number of clicks during the last l impressions by l. Of these impressions, r are in I_j and $l-r$ are in I_{j-1}. Therefore, for j even, the expected number of clicks during the last l impressions is

$$r(1-\epsilon)\lambda + (l-r)\left((1-\epsilon)\lambda+\epsilon\right),$$

and therefore the estimated CTR is the above value plus γ divided by $l+\gamma$ in expectation. When l is large and γ is small, this value is almost always close to its expectation. Therefore, price of a click for this impression is close to pl divided by the above expression. Thus, since the probability of a click for this impression is $(1-\epsilon)\lambda$, the expected payment of the advertiser for this impression can be approximated using the following expression.

$$\frac{pl(1-\epsilon)\lambda}{r(1-\epsilon)\lambda + (l-r)\left((1-\epsilon)\lambda+\epsilon\right)}$$

The average of these values, for all r from 1 to l, can be approximated using the following integral.

$$\frac{1}{l}\int_0^l \frac{pl(1-\epsilon)\lambda}{r(1-\epsilon)\lambda + (l-r)\left((1-\epsilon)\lambda+\epsilon\right)} dr = \frac{p(1-\epsilon)\lambda}{\epsilon}\ln\left(1+\frac{\epsilon}{(1-\epsilon)\lambda}\right).$$

Similarly, the total expected payment of the advertiser in the interval I_j for j odd and $j>1$ can be approximated by the following expression.

$$\frac{p((1-\epsilon)\lambda+\epsilon)}{\epsilon}\ln\left(1+\frac{\epsilon}{(1-\epsilon)\lambda}\right).$$

Denote $\alpha = \frac{\epsilon}{(1-\epsilon)\lambda}$. Therefore, the average payment of the advertiser per impression can be written as follows.

$$p(\frac{1}{2}+\frac{1}{\alpha})\ln(1+\alpha)$$

Since α is a positive constant, it is easy to see that the above expression is strictly greater than p. ■

6 Discussion

In this paper, we discussed pay-per-click marketplaces and proved that a particular class of learning algorithms can reduce click fraud in a simplified setting. Our results lead to several interesting extensions and open questions.

Pay-Per-Acquisition Marketplaces. We focused on pay-per-click marketplaces. Our reasons for this were three-fold: it is a common industry model, it absorbs risk due to market fluctuations for the advertiser, and it simplifies the strategic calculations of the advertiser. The latter two of these comments can be equally well employed to argue the desirability of a pay-per-acquisition marketplace. In these marketplaces, a service provider receives payment from an advertiser only when a click resulted in a purchase. Such systems are used by Amazon, for example, to sell books on web pages: a service provider, say Expedia, can list an Amazon ad for a travel guide with the understanding that, should a user purchase the product advertised, then the service provider will receive a payment. The problem with pay-per-acquisition systems is that the service provider must trust the advertiser to truthfully report those clicks which result in acquisitions. Our results hint at a solution for this problem. We have seen that in a simple scenario with a single ad slot, click-based algorithms are fraud-resistant in the sense that the expected payment per impression of an advertiser can not be *increased* by click fraud schemes. In fact, it can also be shown that this payment can not be *decreased* either. Thus, as click-based learning algorithms reduce fraud in pay-per-click systems, *acquisition-based* learning algorithms induce truthful reporting in pay-per-acquisition systems.

Computational Issues. We have shown that click-based learning algorithms eliminate click fraud. However, in order to be practical and implementable, learning algorithms must also be easily computed with constant memory. The computability of a click-based algorithm depends significantly on the choice of the algorithm. Consider, for example, a simple click-based exponentially-weighted algorithm with $\delta(i) = e^{-\alpha i}$. Just two numbers are needed to compute this estimate: the estimate of the click-through rate for the most recent impression that lead to a click and a counter representing the number of impressions since the last click. However, other click-based algorithms have worse computational issues. Consider an algorithm in which $\delta_i \in \{0, 1\}$ with $\delta_i = 1$ if and only if $i \leq l$ for some (possibly large) l. Then at least l numbers must be recorded to compute this estimate exactly. One interesting question is how efficiently (in terms of the space) a given estimate can be computed.

7 Acknowledgement

We like to thank Omid Etesami and Uriel Feige for fruitful discussions.

References

1. J. Feng, H. K. Bhargava, and D. Pennock. Comparison of allocation rules for paid placement advertising in search engines. In *Proceedings of the fifth International Conference on Electronic Commerce*, Pittsburgh, PA, USA, 2003.
2. J. Goodman. Pay-per-percentage of impressions: an advertising method that is highly robust to fraud. *Workshop on Sponsored Search Auctions*, 2005.
3. `http://adwords.google.com`.
4. Hemant Bhargava Juan Feng and David Pennock. Implementing paid placement in web search engines: Computational evaluation of alternative mechanisms", accepted by informs journal of computing. to appear in the Informs Journal of Computing.
5. D. Mitchell. Click fraud and halli-bloggers. *New York Times*, July 16, 2005.
6. `http://searchmarketing.yahoo.com`.
7. A. Penenberg. Click fraud threatens web. *Wired News*, October 13, 2004.
8. B. Stone. When mice attack: Internet scammers steal money with 'click fraud'. *Newsweek*, January 24, 2005.
9. N. Vidyasagar. India's secret army of online ad 'clickers'. *The Times of India*, May 3, 2004.