Regret-based Online Ranking for a Growing Digital Library*

Erick Delage
Department of Management Sciences
HEC Montréal
Montréal, Qc, Canada
erick.delage@hec.ca

April 27, 2009

Abstract

The most common environment in which ranking is used takes a very specific form. Users sequentially generate queries in a digital library. For each query, ranking is applied to order a set of relevant items from which the user selects his favorite. This is the case when ranking search results for pages on the World Wide Web or for merchandize on an e-commerce site. In this work, we present a new online ranking algorithm, called NoRegret KLRank. Our algorithm is designed to use "clickthrough" information as it is provided by the users to improve future ranking decisions. More importantly, we show that its long term average performance will converge to the best rate achievable by any competing fixed ranking policy selected with the benefit of hindsight. We show how to ensure that this property continues to hold as new items are added to the set thus requiring a richer class of ranking policies. Finally, our empirical results show that, while in some context NoRegret KLRank might be considered conservative, a greedy variant of this algorithm actually outperforms many popular ranking algorithms.

1 Introduction

The most common environment in which ranking is used takes a very specific form. For each query, ranking is applied to order a set of relevant items from which the user selects his favorite. This is for example the case when ranking search results for pages on the World Wide Web or for merchandize on an e-commerce site. In this work, we consider the problem of adapting a ranking mechanism for searches in a digital library using clickthrough feedback as it is generated by the users.

Although many ranking mechanisms have been proposed in the past, very few have addressed this very natural framework. Ranking is typically considered as a learning task which imposes a much different form of training data. In some cases, a relevance score needs to be assigned to

each item (eg. PRank [3]), in other cases, pairwise preference information needs to be provided (eg. RankNet [1]). These methods thus require the learning to be performed offline using a set of hand labeled examples which can be both time consuming and computationally demanding. On the other hand, methods that learn from clickthrough data such as in [7] do not yet offer strong performance guarantees with respect to future users of the search engine. In particular, since, for these methods, the optimal ranking for a new user is uniquely defined given the historical clickthrough data, in principle their is always a possibility that each user will find his preferred item at the bottom of the ranked list.

In this paper, we consider ranking as an online decision problem. At time t, the t-th user generates a query q_t. The system must then make a "non-deterministic ranking decision" on the list $\mathcal{I}_t$ which contains all items related to q_t (*i.e.,* choose a distribution from which to draw a random ranking $\tilde{R}_t$). Given that the item I_t is clicked and assumed to be preferred by the t-th user, the system then incurs a cost, $cost(\tilde{R}_t, I_t)$, modeled as K-L divergence, in compensation for not ranking I_t first with high probability. We propose an adaptive ranking algorithm called NoRegret KLRank which has strong performance guarantees under this framework. More specifically, without making any modeling assumption about the users (*e.g.,* that they are all drawn from the same distribution), the algorithm's long term average cost will converge to the best rate achievable by any competing fixed ranking scheme selected with the benefit of hindsight. We also show how one can ensure that these results hold even in the case where, due to new items being added to the library, one needs to consider a richer set of ranking policies. Effectively, the algorithm will optimally trade off exploration of new ranking policy and exploitation of the best one found so far.

In Section 2, we review related work. In Section 3, we present KLRank's decision model in finer detail. In Section 4, we use the theory of online optimization to derive new results for online ranking. This will involve extending the theory for non-static decision sets. In Section 5, we evaluate our method in an experiment with 22000 queries generated for a set of 9500 scientific articles. Since NoRegret KLRank might be considered a bit conservative in contexts where long term guarantees are not as valuable, we also propose a small variant of this algorithm which outperforms many popular ranking algorithms on this data set.

2 Related Work

Up to now, most studies have focused on learning to rank in an offline fashion. Typically, different forms of training data is used to choose either among a family of relevance classifiers (*e.g.,* in Li *et al.* [10]) or choose among a family of score functions (*e.g.,* Burges *et al.* [1]), ranking is then performed using the best candidate. In [7], Joachims does train an SVM-like model offline using clickthrough data; however, like other offline methods, he offers no performance guarantees with respect to actual users of the system.

More closely related to our work is the work by Crammer *et al.* In [3], they present PRank which holds performance guarantees for relevance classification given that the data is separable. In [4], they address the problem of category ranking through projection which can be adapted to our context. However, without the assumption of a projection that achieves perfect ranking, they are unable to provide any optimality guarantees for their algorithm and limit themselves to bounding worst case loss in a static framework. At a more fundamental level, by considering only deterministic ranking policies, these algorithms are condemned to perform poorly in the

worst case (see Remark 3.1). Our framework considers random ranking policies with worst case "sub-optimality" guarantees.

Recently, Radlinski & Joachims have suggested using a Bayesian approach to model user behavior and devise strategies that exploit clickthrough data more efficiently. More specifically, in both [13] and [14], they make the strong assumption that each user is drawn randomly from a fixed population. In [13], by applying Baye's rule to keep track of a posterior distribution over the true relevance of each item, they are able to propose different heuristics that explore and learn efficiently these values. On the other hand, [14] uses the theory for multi-arm bandit problems to provide performance guarantees for ranking a small library of items given that users are always drawn from the same population. In contrast to their approach, our framework is more realistic as it does not rely on modeling the process generating the sequence of users and allows the continuous addition of items in the library. Our approach also scales easily to large libraries since it relies on extracting features from each item instead of keeping track individual relevance values.

Many of our results are derived using the theory of online convex programming (see Zinkevich [15], and Hazan *et al.* [6]). In this context, the importance of our work relies on formulating a well justified convex ranking problem and deriving new regret bounds which have concrete implications for the task at hand. In fact, the dynamic nature of digital libraries will justify the need to extend the theory to problems with a dynamic feasible set.

3 A Random Ranking Model

We consider the following online ranking framework for a library $\mathcal{L}$ of items $\mathcal{L} = \{I_1, I_2, ..., I_N\}$. An engine accepts queries described by the pair $(q, \mathcal{I})$ where q comes from a set of descriptors Q and $\mathcal{I} \subseteq \mathcal{L}$ is a subset of items considered relevant to q.[1] The ranking algorithm must first choose an ordering R on $\mathcal{I}$. It later learns which item I was found most appealing/interesting/relevant by the user and pay a price of $cost(R, I)$ for not positioning this item first in the list. When no assumption is made about the process generating $(q_t, \mathcal{I}_t, I_t)$ through time, the long term performance of an algorithm that applies deterministic ranking policies can be arbitrarily bad as described in Remark 3.1. For this reason, in order to guarantee worst case performance, we need to consider non-deterministic ranking policies. In what follows, we first introduce a set of random ranking policies, we will then use the notion of K-L divergence to describe a cost function which naturally represent user's satisfaction.

Remark 3.1. : *In principle, a deterministic ranking mechanism, such as one that adapts a score function and ranks from high to low score, can always in the worst case encounter an infinite sequence of users which all find their preferred item at the bottom of the ranked lists, thus achieving the worst possible performance on such a sequence. In practice, unless the users are adversarial, such a scenario should be unlikely to happen. However, this simple fact illustrates how a deterministic mechanism is restricted with respect to the type of guarantees that it can provide and might explain why historically these mechanisms needed to rely on the assumption that future users will behave just like past ones have.*

[1] Instead of being provided by the user, $\mathcal{I}$ could come from a filtering steps on $\mathcal{L}$ given q.

3.1 Non-deterministic Policies for KLRank

We propose a set of non-deterministic ranking policies indexed by $\theta \in \Theta$. A policy $\mu_\theta(\pi; q, \mathcal{I})$ maps a list of items $\mathcal{I}$ and a query q to a distribution over the different permutations functions $\pi : \{1, ..., |\mathcal{I}|\} \to \{1, ..., |\mathcal{I}|\}$. A random ranking created by ordering $\mathcal{I}$ according to $\tilde{\pi}$ will be referred as $\tilde{R}$. Thus, we have that

$$\mathbb{P}_{\mu_\theta}(\tilde{R} = (\mathcal{I}_{\pi(1)}, ..., \mathcal{I}_{\pi(|\mathcal{I}|)}); q, \mathcal{I}) = \mu_\theta(\pi; q, \mathcal{I})$$

which describes the fact that the probability that, given that the query and list of items are q and $\mathcal{I}$ respectively, the mechanism indexed by θ generates the rank $(\mathcal{I}_{\pi(1)}, ..., \mathcal{I}_{\pi(|\mathcal{I}|)})$ with probability $\mu_\theta(\pi; q, \mathcal{I})$.

In order to reduce the policy space to a tractable size and simplify the random sampling process, we only consider random ranking policies that can be expressed in the form

$$\mu_\theta(\pi; q, \mathcal{I}) = \\ f_\theta(\mathcal{I}_{\pi(1)}; q, \mathcal{I}) \prod_{k=2}^{|\mathcal{I}|} f_\theta(\mathcal{I}_{\pi(k)}; q, \mathcal{I} \setminus \{\mathcal{I}_{\pi(1)}, ..., \mathcal{I}_{\pi(k-1)}\}) \quad ,$$

where the parameterized distribution $f_\theta(I; q, \mathcal{I})$ takes the K-logistic form

$$f_\theta(I; q, \mathcal{I}) = \begin{cases} \frac{\exp(s_\theta(q,I))}{\sum_{k=1}^{|\mathcal{I}|} \exp(s_\theta(q,\mathcal{I}_k))} & \text{if } I \in \mathcal{I} \\ 0 & \text{o.w.} \end{cases} ,$$

with $s_\theta(q, I) = \Phi(q, I)^\mathsf{T}\theta$ for some feature extracting function $\Phi(q, I) : Q \times \mathcal{L} \to \Re^n$ and some $\theta \in \Theta \subseteq \Re^n$. The forms of Θ and $\Phi(q, I)$ depend heavily on the application at hand.

An interesting property of ranking according to $\mu_\theta(\pi; q, \mathcal{I})$ is that sampling a ranking from this distribution can easily be performed by drawing sequentially each item from highest to lowest rank according to $f_\theta(I; q, \mathcal{I}')$ which depends on the set of remaining unordered items $\mathcal{I}'$. More specifically, letting $\tilde{R}_k$ be the k-th item in the random order $\tilde{R}$,

$$\mathbb{P}_{\mu_\theta}(\tilde{R}_{k+1} = \mathcal{I}_{\pi(k+1)} | \tilde{R}_{1:k} = (\mathcal{I}_{\pi(1)}, ..., \mathcal{I}_{\pi(k)}); q, \mathcal{I}) \\ = f_\theta(\mathcal{I}_{\pi(k+1)}; q, \mathcal{I} \setminus \{\mathcal{I}_{\pi(1)}, ..., \mathcal{I}_{\pi(k)}\}) \quad .$$

This definition also leads to a proper distributions for $\tilde{R}$.

An interesting property of each policy in this class is its consistency with respect to the probability of ranking I_1 higher than I_2, under fixed q, for different choices of $\mathcal{I}$ that contain both items. More specifically, for all $\mathcal{I} \supseteq \{I_1, I_2\}$, we have that :

$$\mathbb{P}_{\mu_\theta}(\tilde{\pi}^{-1}(I_1) > \tilde{\pi}^{-1}(I_2); q, \mathcal{I}) = \\ \mathbb{P}_{\mu_\theta}(\tilde{\pi}^{-1}(I_1) > \tilde{\pi}^{-1}(I_2); q, \{I_1, I_2\}) ,$$

where $\tilde{\pi}^{-1}(\cdot)$ is the random inverse mapping of $\tilde{\pi}(\cdot)$ such that $k = \tilde{\pi}^{-1}(\tilde{\pi}(k))$ (see Appendix A for proof). This property is a natural and popular one for a ranking policy as it guarantees that prioritizing I_1 over I_2 only depends on q and not on the set $\mathcal{I}$ that is considered. For instance, it is satisfied by a ranking mechanism that orders item according to how relevant they are to the query.

3.2 K-L Divergence based Cost for KL Rank

Now that we have described a set of parameterized random ranking policies, we are interested in deriving a meaningful performance measure. We assumed earlier that the ranking mechanism is informed of the item I_t chosen by the user that generated query q_t. In fact, this information tells us that according to user t, the best ranking policy should position I_t first. Given that we cannot infer other characteristics of user t's preferred policy, it is natural to model his satisfaction (our cost) as the Kullback-Leibler divergence between the applied random ranking policy and the set $\mathcal{D}_t$ of distributions over rankings that fits this preference information:

$$\mathcal{D}_t = \left\{ \nu : \Pi \to \Re \;\middle|\; \begin{array}{l} \nu(\pi) \geq 0 \quad \forall \pi \in \Pi \\ \sum_{\pi \in \Pi} \nu(\pi) = 1 \\ \mathbb{P}_\nu(\tilde{R}_1 = I_t) = 1 \end{array} \right\} \quad .$$

Given a set of query-item pairs (q_t, I_t), we therefore aim at finding θ which minimizes the average "K-L cost" (or dissatisfaction). Thus, an optimal ranking policy is a minimizer of

$$\underset{\theta \in \Theta}{\text{minimize}} \qquad \frac{1}{T} \sum_{t=1}^{T} \min_{\nu \in \mathcal{D}_t} D_{\text{KL}} \left(\nu(\pi) \| \mu_\theta(\pi; q_t, \mathcal{I}_t) \right) \quad ,$$

where $D_{\text{KL}} \left(\nu(\pi) \| \mu(\pi) \right) = \sum_{\pi \in \Pi} \nu(\pi) \log \left(\frac{\nu(\pi)}{\mu(\pi)} \right)$.

Remark 3.2. : *We believe that KL-cost is a sound way of measuring satisfaction of a user that interact with a non-deterministic ranking mechanism. Unlike measures that are simply based on the rank of the clicked item, KL-cost penalizes directly the fact that the item was unlikely to appear early in the list.*

By exploiting the properties of K-L divergence and the structure of any policy parameterized by θ, the cost function can be reduced to a more convenient form:

$$\begin{aligned} cost(\theta; q_t, \mathcal{I}_t, I_t) &:= \min_{\nu \in \mathcal{D}_t} D_{\text{KL}} \left(\nu(\pi) \| \mu_t(\pi) \right) \\ &= \min_{\nu \in \mathcal{D}_t} D_{\text{KL}} \left(\mathbb{P}_\nu(\tilde{R}_1 = I_t) \middle\| \mathbb{P}_{\mu_t}(\tilde{R}_1 = I_t) \right) \\ &\qquad + \mathbb{E}_\nu \left[D_{\text{KL}} \left(\nu(\pi | \tilde{R}_1 = I_t) \middle\| \mu_t(\pi | \tilde{R}_1 = I_t) \right) \right] \\ &= -\log(\mathbb{P}_{\mu_t}(R_1 = I_t)) \\ &\qquad + \min_{\nu \in \mathcal{D}_t} \mathbb{E}_\nu \left[D_{\text{KL}} \left(\nu(\tilde{R} | \tilde{R}_1 = I_t) \middle\| \mu_t(\tilde{R} | \tilde{R}_1 = I_t) \right) \right] \\ &= -\log(\mathbb{P}_{\mu_t}(R_1 = I_t)) + 0 \\ &= -s_\theta(q_t, I_t) + \log \left(\sum_{I \in \mathcal{I}_t} \exp(s_\theta(q_t, I)) \right) \quad , \end{aligned}$$

where $\mu_t(\pi)$ stands short for $\mu_\theta(\pi; q_t, \mathcal{I}_t)$.

The ranking decision problem thus takes the convex form:

$$\min_{\theta \in \Theta} \sum_{t=1}^{T} -\Phi(q_t, I_t)^\mathsf{T} \theta + \log \left(\sum_{I \in \mathcal{I}_t} \exp(\Phi(q_t, I)^\mathsf{T} \theta) \right) . \tag{1}$$

Convexity of this objective function can be verified through the Hessian $\nabla^2 cost(\theta; q, \mathcal{I}, I) \succeq 0$. One can first derive the gradient of $cost(\theta; q, \mathcal{I}, I)$ to be

$$\nabla cost(\theta; q, \mathcal{I}, I) = -(\Phi(q, I) - \mathbb{E}_{\mu_\theta}[\Phi(q, \tilde{R}_1)]) \ ,$$

while its Hessian matrix takes the form:

$$\begin{aligned}\nabla^2 cost(\theta; q, \mathcal{I}, I) = \mathbb{E}_{\mu_\theta}[\Phi(q, \tilde{R}_1)\Phi(q, \tilde{R}_1)^\mathsf{T}] \\ - \mathbb{E}_{\mu_\theta}[\Phi(q, \tilde{R}_1)]\mathbb{E}_{\mu_\theta}[\Phi(q, \tilde{R}_1)]^\mathsf{T} \succeq 0 \ ,\end{aligned}$$

where the positive semi-definiteness of the Hessian matrix is assured by the fact that it is the covariance matrix of the random vector $\Phi(q, \tilde{R}_1)$.

The optimization model that is presented in this section for choosing a non-deterministic ranking policy based on clickthrough data is new and could be solved offline to find the policy that performs best on the data.[2] In what follows, we will actually propose an online mechanism that adapts its policy has it interacts with the users, yet performs ultimately as well as a policy obtained by solving the model offline with the same set of users.

Remark 3.3. : *Note that in [1], the authors presented an offline ranking model that is related to ours but which attempts to best fit some assumed prior knowledge of users' pairwise preferences. Our model has a more general form since it can handle pairwise preferences as a special case. It is also better justified for learning from clickthrough data since, in this context, the raw information takes the form of choosing one item among a set of items, which size is typically larger than two.*

4 Regret in Online Ranking

We are interested in studying the online adaptation of a random ranking mechanism of the form presented in Section 3. We formulate this problem in the framework of online convex optimization as presented in [15] and [6], where the long term regret of an online strategy A, that adaptively chooses $\theta_t \in \Theta$, can be defined as:

$$\mathcal{R}_A(T) = \sum_{t=1}^{T} cost(\theta_t; q_t, \mathcal{I}_t, I_t) - \min_{\theta \in \Theta} \sum_{t=1}^{T} cost(\theta; q_t, \mathcal{I}_t, I_t) \ .$$

This measure actually compares the total performance attained by the adaptive ranking algorithm A to the performance of the best fixed random ranking policy, in the set of policies indexed by $\theta \in \Theta$, chosen with full knowledge of $\{q_t, \mathcal{I}_t, I_t\}$. We first address the case where the time horizon is known and $\mathcal{L}$ and Q are fully determined. Later, we will extend our analysis to an undefined horizon using the doubling trick. We will see however that, in order to be realistic, the analysis of cumulated regret needs to account for the dynamic nature of digital libraries. Specifically, as the library grows one necessarily needs to resolve how to enrich the set of ranking policies. In fact, the solution we propose is relevant in the context of developing online version for many rank learning models as will be discussed in Section 5.1.

[2] In practice, as long as the subsets of items are not too large a solution to Problem 1 can be found efficiently using a truncated Newton method to compute search directions (see [5] for more details on this method).

4.1 Finite Horizon Bounds for a Static Library

We consider a finite time horizon of T and full knowledge about the members of the library $\mathcal{L}$ and the query set Q. Based on the choice of Θ and $\Phi(\cdot)$, let's first define the following constants:

$$\begin{aligned} F &= \max_{q\in Q, I\in\mathcal{L}} \|\Phi(q,I)\| \qquad\qquad D = \max_{\theta_1,\theta_2\in\Theta} \|\theta_1-\theta_2\| \\ G &= \max_{q\in Q,\mathcal{I}\subset\mathcal{L},I\in\mathcal{I},\theta\in\Theta} \|\nabla cost(\theta;q,\mathcal{I},I)\| \\ &= \max_{q\in Q,\mathcal{I}\subset\mathcal{L},I\in\mathcal{I},\theta\in\Theta} \|\Phi(q,I)-\mathbb{E}_{\mu_\theta}[\Phi(q,\tilde{R}_1)]\| \le 2F \\ H &= \min_{q\in Q,\mathcal{I}\subset\mathcal{L},I\in\mathcal{I},\theta\in\Theta} \min_{\{x\in\Re^n|\|x\|\le 1\}} x^{\mathsf{T}}\mathrm{Cov}_{\mu_\theta}[\Phi(q,\tilde{R}_1)]x \end{aligned}$$

Based on the work by Hazan *et al.* [6], we can already derive an algorithm which is known to achieve logarithmic regret.

Proposition 4.1. : *(Hazan et al. (2007)) Applying the Online Gradient Descent method proposed in [6] leads to a regret bound of :*

$$\mathcal{R}_{OGD}(T) \le \frac{G^2}{2H}(1+\log(T)) \ .$$

Although this result is appealing, we expect the proposed algorithm to actually perform poorly in many applications due to $\frac{G^2}{H}$ taking a disproportioned value as shown in the following example. In the remainder of the paper, we will re-visit this Practical Example at different occasion to illustrate the implications of our theoretical results, while Section 5 will use it to compare performance of different learning algorithms.

Practical Example (Part 1.) : *Let's consider the case where one wishes that each item be associated with its own fitness parameter θ_i and that the random ranking be generated using*

$$f_\theta(I_i;q,\mathcal{I}) \ = \ \frac{\exp(\theta_i)}{\sum_{k:I_k\in\mathcal{I}}\exp(\theta_k)} \ , \ \forall \ I_i\in\mathcal{I} \ .$$

This can be done by defining $\Phi(q,I)\in\Re^n$ with $n=N$, such that $\Phi(q,I_i)=e_i$, where e_i is the i-th column of the $N\times N$ identity matrix. One can also consider Θ to be a ball of radius r centered at 0. Then, it is the case that $D=2r$, $F=1$, $G=2$ and more importantly that $H<1/(|\mathcal{L}|\exp(r))$ since

$$\frac{\partial^2}{\partial\theta_1^2} cost(\theta;q,\mathcal{L},I_1) = \frac{1}{|\mathcal{L}|\exp(r)}\left(1-\frac{1}{|\mathcal{L}|\exp(r)}\right) \ ,$$

for $\theta_1=-r$ and $\theta_i=0$ for $i\ge 2$. Thus, the Online Gradient Descent method has a regret bound of an order larger than $O(4|\mathcal{L}|\exp(r)\log(T))$ and is therefore not practical for practical sizes of r. In fact, a similar argument can be made for the Online Newton Step algorithm proposed in [6]. This leads us to believe that methods which perform logarithmic regret cannot be practically applied to this problem.

Table 1: Greedy Projection with Exponential Restart Algorithm

(GPER)

Let $t = 0$, for cycles $m = 1, 2, ...$:

- Initialize $\theta_{t+1} \in \Theta_m$ and compute $\alpha = D_m/(2\sqrt{2}F_m)$
- For $k = 1, 2, ..., 2^{m-1}$:
  1. At time $t = 2^{m-1} + k - 1$, receive query q_t over set of items $\mathcal{I}_t$
  2. Generate a random ranking from $\tilde{R}$ from $\mu_\theta(\pi; q_t, \mathcal{I}_t)$
  3. Learn about the preferred item I_t
  4. Update $\theta_{t+1} = \mathcal{P}_{\Theta_m}\left(\theta_t + \frac{\alpha}{k^{1/2}} \nabla cost(\theta_t; q_t, \mathcal{I}_t, I_t, \Phi_t(\cdot))\right)$

This example justifies considering an algorithm such as the Greedy Projection (GP) algorithm presented in [15] which considers at time t taking the step

$$\theta_{t+1} = \mathcal{P}_\Theta\left(\theta_t - \eta_t \nabla cost(\theta_t; q_t, \mathcal{I}_t, I_t)\right) ,$$

where $\mathcal{P}_\Theta(\theta)$ is the projection of θ on Θ and η_t is a step size. As demonstrated in [15], this method is guaranteed to lead to regret being bounded by $O(\sqrt{T})$. Although average regret decays more slowly with the GP algorithm, we are about to show that its performance bound is much less sensitive to design choices. In fact, our version of the GP algorithm guarantees sub-linear regret even when Θ and $\Phi(\cdot)$ have size comparable to T. We believe this property is necessary when working with libraries that have widely diversified content: e.g., the World Wide Web.

Theorem 4.2. : *Given that $DF \leq KT^{1/4}$ for some $K > 0$ and that $\eta_t = \frac{D}{2\sqrt{2}F} t^{-1/2}$, then the regret cumulated by the Greedy Projection algorithm is $\mathcal{R}_{GP}(T) \leq 2\sqrt{2}KT^{3/4}$. In particular, average cumulated regret is driven to zero at the rate of $O(T^{-1/4})$.*

Proof. This result is derived using similar arguments to those used by Zinkevich to prove Theorem 1 in [15]. In his proof, Zinkevich's demonstrates that for any sequence $(\eta_1, \eta_2, ..., \eta_T)$, the regret of the Greedy Projection algorithm applied to an online convex programming problem of the type $\text{minimize}_{\theta \in \Theta} \sum_{t=1}^{T} cost(\theta_t; q_t, \mathcal{I}_t, I_t)$ will cumulate a regret bounded above by $\mathcal{R}_{GP}(T) \leq \frac{D^2}{2\eta_T} + \frac{G^2}{2} \sum_{t=1}^{T} \eta_t$. Instead of choosing $\eta_t = t^{-1/2}$, our result relies on defining $\eta_t = \frac{D}{2\sqrt{2}F} t^{-1/2}$ and deriving new interesting conclusions.

$$\begin{aligned} \mathcal{R}_{GP}(T) &\leq \frac{D^2}{2}\frac{2\sqrt{2}F}{D}\sqrt{T} + \frac{4F^2}{2}\frac{D}{2\sqrt{2}F}(2\sqrt{T} - 1) \\ &\leq 2\sqrt{2}DF\sqrt{T} \leq 2\sqrt{2}KT^{3/4} , \end{aligned}$$

where we used the fact that $\sum_{t=1}^{T} \frac{1}{\sqrt{t}} \leq 1 + \int_{t=1}^{T} \frac{1}{\sqrt{t}} dt = 2\sqrt{T} - 1$. □

Note that based on [15], we could already have concluded that for $D \leq K_1 T^{1/8}$ and $F \leq K_2 T^{1/8}$ a step size of $\eta_t = 1/\sqrt{t}$ achieves a regret bounded by $O\left((K_1^2 + K_2^2)T^{3/4}\right)$. However, with our refined step size, we are able to tighten this bound and consider the scale of D and F jointly.

Practical Example (Part 2.) : *When applied to our example, given that $r = \alpha T^{1/4}$ for some $\alpha > 0$, since $F = 1$ our new bound leads to $\frac{1}{T}\mathcal{R}_{GP}(T) \leq 4\sqrt{2}\alpha T^{-1/4}$ which directly relates the range of competing random ranking policies to our aptitude to learn how to best satisfy T users of our system.*

4.2 Long Term Bounds for Dynamic Libraries

In the case where T is not determined, we need to take into account the dynamics of the library and query set. In order to stay competitive, it is also necessary that our algorithm allows new feature functions to be added in order to better distinguish each item as the total number increases. Obviously, the mechanism's performance depends heavily on the choice of features. However, the problem of selecting good features falls outside the scope of this work. For this reason, we chose to compare performance to the best achievable using the available set of features. The ranking problem now takes the following form. Can a ranking mechanism adjust the weights θ_t in order to perform adequate ranking in the dynamic environment parameterized by $\{\mathcal{L}_t, Q_t, \Phi_t(\cdot)\}$?

For simplicity, we start by imposing a maximum number n of features returned by $\Phi_t(\cdot)$ but the theory holds for n arbitrarily large. We also make the following necessary assumption.

Assumption 4.3. : *Let $(\Theta_1, \Theta_2, ...)$ be a sequence of feasible sets such that for any $m > 0$, one is assured that for some $K \geq 0$ it is the case that $D_m F_m < K 2^{m/4}$ where*

$$D_m = \max_{x,y \in \Theta_m} \|x - y\| \qquad \& \qquad F_m = \max_{t < 2^m} \max_{q \in Q_t, I \in \mathcal{L}_t} \|\Phi_t(q, I)\| .$$

Intuitively, although one does not know at time 0 what features he will be using in the long run, one should be able at time $t = 2^{m-1}$ to commit to some $\Phi_t(\cdot)$ and Θ_m for which he knows that $D_m F_m < K 2^{m/4}$ holds. This assumption allows us to focus on comparing the performance of the online mechanism, for times in the range $2^{m-1} \leq t < 2^m$, to the set of ranking mechanisms with index $\theta \in \Theta_m$. In this context, long term regret should be measured as

$$\mathcal{R}_A(T) = \sum_{t=1}^{T} cost(\theta_t; q_t, \mathcal{I}_t, I_t, \Phi_t(\cdot)) - \min_{\theta \in \Re^n} \sum_{t=1}^{T} cost(\mathcal{P}_{\Theta_{\phi(t)}}(\theta), q_t, \mathcal{I}_t, I_t, \Phi_t(\cdot)) \quad , \tag{2}$$

where $\phi(t) = \min\{m | t < 2^m\}$. In simple terms, $\mathcal{R}_A(T)$ compares the performance of the adaptive policy $\theta_t \in \Theta_{\phi(t)}$ to a fixed policy $\theta \in \Re^n$ selected with full knowledge of $\{q_t, \mathcal{I}_t, I_t, \Phi_t(\cdot)\}$ and applied after being projected on $\Theta_{\phi(t)}$ considered feasible at time t.

We now consider an online algorithm, inspired by applying the doubling trick (see [2] for details), which we call Greedy Projection with Exponential Restart (see Table 1).

Theorem 4.4. : *Given an arbitrary sequence of queries* $((q_1, \mathcal{I}_1, \Phi_1(\cdot)), (q_2, \mathcal{I}_2, \Phi_2(\cdot)), ..., (q_T, \mathcal{I}_T, \Phi_T(\cdot)))$ *and its associated sequence of preferred top items* $(I_1, I_2, ..., I_T)$*, let* $(\Theta_1, \Theta_2, ...)$ *be a sequence of feasible sets that satisfy Assumption 4.3. Then, the regret, as defined in Equation 2, that is cumulated by the GPER algorithm is bounded by* $\mathcal{R}_{GPER}(T) \leq 10K(T^{3/4} + 1)$*. In particular, average cumulated regret converges to zero at the rate of* $O(T^{-1/4})$*.*

The proof of this result is deferred to Appendix B. Overall, this theorem provides us with strong guarantees that, on a large enough horizon T, the GPER algorithm performs as well as a fixed ranking policy chosen in hindsight. The result also provides us with useful guidelines for adjusting the mechanism, through either $\Phi_t(\cdot)$ or Θ_m, without affecting long term performances.

Practical Example (Part 3.) : *In our example, we can consider that items are indexed chronologically* $(I_1, I_2, ...)$ *as they are added to the library. Item i is associated with a fitness parameter* θ_i*. Considering that we are interested in* $\Phi_t(q_t, I_i) = e_i$ *as in the static case, this implies that each time a item is added to the library a new fitness feature needs to be experimented with by the ranking mechanism. Because by definition* $F_m = \max_i \|e_i\| = 1$ *for all m, we are left with the following question. How quickly should we allow* Θ_m *to grow in "volume" as the library grows? A any given time, having a larger set* Θ_m *allows more fitness parameters to take on values away from zero thus allowing the ranking to be more informed. In order to preserve the regret guaranty expressed in Theorem 4.4, we must choose* $\Theta_m = \{\theta | \|\theta\| \leq \alpha T_m^{1/4}\}$ *so that the resulting mechanism still satisfies Assumption 4.3 with* $K = 2\alpha$ *since* $D_m F_m = 2\alpha 2^{m/4} = K2^{m/4}$*. Effectively, we have allowed* Θ_m *to grow as the library grows, thus allowing more fitness parameters to take on non-zero values and influence the ranking. Applying the GPER algorithm is therefore guaranteed to lead to a cumulated regret of* $\mathcal{R}_{GPER}(T) \leq 20\alpha(T^{3/4} + 1)$*. Thus, the GPER algorithm generates average regret that decreases to zero at the rate of* $O(T^{-1/4})$*. Note that we constructed* Θ_m *such that it grows at a rate of* $T^{1/4}$*; thus, in practice this suggests that items should be added at a rate smaller than* $\sqrt{T}$ *in order to expect good ranking performance. Otherwise, the low regret guarantees effectively ends up requiring an increasingly restrictive set of policies.*

5 Experiments on a Growing Database of Articles

In this section, we compare how different online ranking algorithms perform for the task of ordering scientific papers for users, each interested in a different topic. Our experiments use a citation data set named Cora, created by A. McCallum [11], containing data on more than 9500 papers in the field of computer science. In Cora, each paper is associated with its year of publication, its list of citations, and its most relevant topic. In order to simulate the process of users querying a growing library of papers before choosing their preferred one, we create a natural $\{\mathcal{L}_t, q_t, \mathcal{I}_t, \Phi_t(\cdot), I_t\}$ sequence from the Cora data set. The papers are first ordered chronologically. Then, given the k-th published paper, each of its citations is considered as a

Over 22000 queries	Avg K-L Cost	Avg Rel. Click Dist.	Clicked 1st Rate	Avg NDCG score
NoRegret KLRank	4.63 (1)	25.7% (4)	14.3% (5)	34.4% (5)
Greedy KLRank	∞ (3)	19.2% (1)	17.8% (1)	39.1% (1)
Online RankNet	∞ (3)	19.8% (2)	16.1% (4)	37.4% (2)
Category Ranking	∞ (3)	23.2% (3)	16.5% (2)	37.3% (2)
HITS / PageRank	∞ (3)	26.5% (5)	16.6% (2)	36.2% (4)
McRank	∞ (3)	26.3% (5)	14.0% (5)	34.1% (5)
Random Ranking	4.93 (2)	45.0% (7)	9.7% (7)	27.6% (7)

Table 2: Performance comparison on the Cora data set : (i) average K-L cost, (ii) average relative distance of click in ordered lists, (iii) rate at which top item is clicked, and (iv) average NDCG score. Columns also present how algorithms ranked, from best (1) to worst (7), with respect to the metric.

query to the library of previously published papers $\mathcal{L}_t$: q_t being the topic of the cited paper, $\mathcal{I}_t$ being the set of papers in $\mathcal{L}_t$ on this topic, and I_t being the cited paper itself. The algorithms are then evaluated with respect to the rank assigned to I_t.

5.1 Algorithms under Evaluation

We evaluated the ranking algorithms listed below. In order to focus on their capacity to learn good rankings, all our implementations used the same feature extraction functions: $\Phi_t(q, I_i) = e_i$, which was the case considered in the set of practical examples. Note that because we expect NoRegret KLRank to act a bit conservatively in some applications, we introduce Greedy KL-Rank which implements a simple opportunistic variant to the original algorithm. The new results we derived for the regret cumulated in a ranking framework also allow us to derive an original online ranking version of the RankNet model presented in [1], which we refer as Online RankNet.

NoRegret KLRank We applied the GPER algorithm presented in our practical example with $\alpha = 10$. Based on the results presented in Part 3 of our Practical Example, we know that after T iterations this method is guaranteed to perform ϵ-optimally, for $\epsilon = 200(T^{-1/4} + 1/T)$, in terms of average K-L cost when compared to any fixed random ranking policy.

Greedy KLRank Because NoRegret KLRank might perform conservatively in practice, we evaluate the performance of ranking the items directly according to $s_\theta(q, I_i) = \theta_i$ instead of randomly from $\mu_\theta(\pi; q, \mathcal{I})$. Although this method is not known to cumulate low average regret with respect to K-L cost, using Theorem 4.4 that the parameter that is learned using GPER will achieve low average regret in terms of long term "fitness" to the clickthrough data for the

loss function defined as

$$loss(\theta; q_t, \mathcal{I}_t, I_t) = \\ -\Phi(q_t, I_t)^\mathsf{T}\theta + \log\left(\sum_{I\in\mathcal{I}_t} \exp(\Phi(q_t, I)^\mathsf{T}\theta)\right).$$

Online RankNet We significantly adapt the RankNet algorithm to fit our framework by interpreting the user feedback as a source of pairwise information about the preferred ranking. More specifically, we define a new loss function,

$$loss(\theta; q_t, \mathcal{I}_t, I_t) = \frac{1}{|\mathcal{I}_t| - 1} \sum_{I\in\mathcal{I}_t\setminus I_t} \Big(-\Phi(q_t, I_t)^\mathsf{T}\theta \\ + \log\Big(\exp(\Phi(q_t, I_t)^\mathsf{T}\theta) + \exp(\Phi(q_t, I)^\mathsf{T}\theta)\Big)\Big) \quad ,$$

where the normalization ensures that the weight of each time step's contribution is independent of $|\mathcal{I}|$ and again that

$$\max_{q\in Q, \mathcal{I}\subset\mathcal{L}, I\in\mathcal{I}, \theta\in\Theta} \|\nabla loss(\theta; q, \mathcal{I}, I)\| \leq 2 \max_{q\in Q, I\in\mathcal{L}} \|\Phi(q, I)\| .$$

We consider an online version of this model which is updated using our GPER algorithm. The algorithm finally ranks items directly with θ_t. Again, Theorem 4.4 can be used to guarantee no long term average regret with respect to the cumulated loss in "fitness" of the sequence of parameters $\{\theta_t\}_{t=1}^T$: $\sum_{t=1}^T loss(\theta_t; q_t, \mathcal{I}_t, I_t)$.

Category Ranking We used the method presented in [4] to perform online ranking on items instead of categories. This method guarantees a worst case loss bound on the sum of ranking distances; however, it does not bound any form of cumulated regret.

HITS & PageRank Based on Kleinberg's arguments in [9], the level of "authority" of a node of the citation graph should indicate its value as a citation. Given that $A_t \in \Re^{N\times N}$ is a matrix with $A_t(i, j) = 1$ if article i cited j prior to time t, the theory suggests ranking according to a score vector $x \geq 0$ which is a left eigenvector of $A^\mathsf{T}A$ with largest eigenvalue. PageRank, found in [12], can also be applied by considering a random surfer on the transition matrix A with random restarts. In practice, we believe both methods performed similarly since they rely on the similar information extracted from the evolving graph of citations.

2-class McRank Classification methods have been very effective at predicting the opinion of users in a framework that is driven by user ratings. Unfortunately, when adapted to a framework driven by the ordering of items, they can fail to distinguish what makes items more popular than others. We choose McRank as an example. The most natural application considers a 2-class problem where items that should be ranked first are part of class 1, while other items

are in class 0. To learn the two classes, we use the logistic regression model with the popular log-likelihood loss function:

$$loss(\theta; q_t, \mathcal{I}_t, I_t) = -\log(\mathbb{P}_\theta(\text{Rank}(I_t) = 1; q_t)) - \sum_{I \in \mathcal{I}_t / I_t} \log(\mathbb{P}_\theta(\text{Rank}(I) \neq 1; q_t)) \ ,$$

where

$$\mathbb{P}_\theta(\text{Rank}(I_i) = 1; q) = 1/(1 + \exp(-\Phi(q, I_i)^T \theta)) \ .$$

When this model is learned offline, the ranking algorithm actually reduces to sorting the items according to the empirical proportion of time that each item was clicked on when it appeared in the list: $\theta_i = -\log(\hat{p}_i^{-1} - 1)$ with $\hat{p}_i = \frac{\sum_t \mathbb{1}\{I_t = I_i\}}{\sum_t \mathbb{1}\{I_i \in \mathcal{I}_t\}}$.

Random Ranking We use as a reference the uniform distribution over rankings.

Remark 5.1. : *We wish to emphasize the distinction between the concept of a loss function, used in describing Greedy KLRank and Online Ranknet, and the concept of the KL-cost. In the first case, the problem is formulated strictly in terms of finding parameters θ that explains the clickthrough data appropriately with the hope that such θ will lead to good scores to use in ranking. On the other hand, KL-cost evaluates the ranking policies directly with an explicit measure of satisfaction for the users. For this reason, we believe the regret bound for NoRegret KLRank can be interpreted as a guarantee on worst case revenues generated by the search engine.*

5.2 Results & Discussion

Table 2 presents the performance of the different ranking algorithms over 22000 queries created from the Cora data set. In this table, the first column expresses the average K-L cost experienced by the different algorithms. The second column compares the average relative distance of the "clicked" items in the ordered lists. The third column compares the proportion of queries which ended up selecting the top ranked item. The final column presents a popular ranking metric called average Normalized Discounted Cumulated Gain: $\text{NDCG}_t = \log(2)/\log(1 + \text{Dist}(I_t))$ where $\text{Dist}(\cdot)$ refers to distance in the ordered list (see [10] for details on this metric).

Table 2 reveals us that NoRegret KLRank performed a bit conservatively on this data set. Although it does achieve lowest K-L cost, it ends up suffering performance loss when compared to Category Ranking, Online RankNet, and Greedy KLRank. We consider however that in many applications, a 5% difference in relative click distance, for instance, might be a small price to pay in order to provide the strong a priori guarantee provided by Theorem 4.4. While other methods could be unreliable in a noisy or even adversarial environment, we are assured with NoRegret KLRank to be consistent in terms of achieving optimal long-term KL-cost.

Considering applications where the guarantees provided by NoRegret KLRank are less valuable, we remark that the two proposed heuristic, Greedy KLRank and Online Ranknet, can provide good alternatives. Although Greedy KLRank as a weaker worst case performance guarantee, it appear to be more opportunistic with this particular data set resulting in significant gains in performance. In fact, our experimental results indicate that it outperformed all other methods for most of the metrics considered on the Cora data set. The results also seem to

indicate that the loss function associated with Greedy KLRank is better suited for learning from clickthrough data then the one used by Online Ranknet.

6 Conclusion

In this paper, we considered the problem of using clickthrough data as it is generated by the user to adapt a ranking policy. We first argued that non-deterministic ranking policies needed to be used in order to have performance guarantees that are independent of how users visit the engine. After modeling user satisfaction using the KL-cost, we extended the theory of regret minimization so that it could accommodate a constraint that is critical to the task of ranking: i.e., the fact that as time goes by digital libraries get more populated and diverse. Our analysis allowed us to derive specific recommendations on how new features should be added and the parameter space increased in order to allow for richer policies. Based on these recommendations, we could guarantee that the GPER algorithm would learn to use the newly available policies profitably. The newly found properties of GPER led to the description of three new online ranking algorithms which were evaluated at the task of ranking queries on a growing database of articles. We believe that these methods benefited heavily from adding new features as recommended by the theory. Although NoRegret KLRank has strong performance guarantees, our experiments indicate that it can act a bit conservatively. On the other hand, Greedy KLRank outperformed all other methods at this task. Future work on this subject will consist of confirming these observations using other data sets.

Although discovering the "true" relevance of each item is not the main objective of this mechanism, NoRegret KLRank's will converge to a ranking that reflects the "true" relevance of each item given that each user click reports this relevance truthfully. In practice, this might not be the case, yet we believe this mechanism is still a promising heuristic to use. Joachims *et al.* [8] have shown that clickthrough data can be biased towards items that have higher ranks. Although our mechanism does not account for this bias in its current form, we expect it to be less sensitive to it since, by randomly generating a ranking for each user, it provides the opportunity to more items to take on the top positions. In some context, it is also the case that some users have personal interests in influencing the ranking with their clicks. Being a robust framework by nature, we expect NoRegret KLRank to be resistant to these attacks given that the proportion of malignant clicks is kept small enough. Overall, these remain two important open questions which should be investigated deeper.

7 Acknowledgments

The author acknowledge the Fonds Québécois de la recherche sur la nature et les technologies for its financial support. He also wishes to thank Ramesh Johari for valuable discussions on the subject.

References

[1] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proc. of 22nd Intl. Conf. on Machine Learning*,

pages 89–96, 2005.

[2] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[3] K. Crammer and Y. Singer. Pranking with ranking. In *Adv. in Neural Information Processing Systems*, volume 14, pages 641–647, 2001.

[4] K. Crammer and Y. Singer. Loss bounds for online category ranking. In *18th Annual Conf. on Learning Theory*, pages 48–62, 2005.

[5] R. Dembo and T. Steihaug. Truncated-newton algorithms for large-scale unconstrained optimization. *Mathematical Programming*, 26:190–212, 1983.

[6] E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

[7] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. of the 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 133–142, 2002.

[8] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proc. of 28th Annual Intl. ACM SIGIR Conf. on Research and Dev. in Information Retrieval*, pages 154–161, 2005.

[9] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[10] P. Li, C. Burges, and Q. Wu. McRank: Learning to rank using multiple classification and gradient boosting. In *Adv. in Neural Information Processing Systems*, pages 897–904. 2008.

[11] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000. Cora data set can be found at : www.cs.umass.edu/∼mccallum/code-data.html.

[12] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[13] F. Radlinski and T. Joachims. Active exploration for learning rankings from clickthrough data. In *Proc. of the 13th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 570–579, 2007.

[14] F. Radlinski, R. Kleinberg, and J. Thorsten. Learning diverse rankings with multi-armed bandits. In *Proc. of 25th Intl. Conf. on Machine Learning*, pages 784–791, 2008.

[15] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proc. of 20th Intl. Conf. on Machine Learning*, pages 928–936, 2003.

A Consistency of KLRank Model

Proposition A.1. : *The KLRank model has the following property:* $\forall\ \mathcal{I} \supseteq \{I_1, I_2\}$,

$$\mathbb{P}(\tilde{\pi}^{-1}(I_1) > \tilde{\pi}^{-1}(I_2); q, \mathcal{I}) = \mathbb{P}(\tilde{\pi}^{-1}(I_1) > \tilde{\pi}^{-1}(I_2); q, \{I_1, I_2\}),$$

where the probability is measured according to $\mu_\theta(\pi; q, \mathcal{I})$ *from which the random mapping* $\tilde{\pi}$ *(and its associated random inverse mapping* $\tilde{\pi}^{-1}$*) is generated.*

Proof. By induction on the items in $\mathcal{I} = \{I_1, I_2, ..., I_{|\mathcal{I}|}\}$, if $\mathcal{I} = \{I_1, I_2\}$ then the result holds by the definition of $\mathbb{P}(\tilde{R}; q, \mathcal{I}))$. Now assuming that the result holds for any subset of size $m-1$ that contains I_1 and I_2 then the result also holds for any subset $\mathcal{I}'$ of size m which also contains them by the following argument.

$$\begin{aligned}
&\mathbb{P}(\tilde{\pi}^{-1}(I_1) > \tilde{\pi}^{-1}(I_2); \mathcal{I}') \\
&\quad = \ \mathbb{P}(\tilde{R}_1 = I_1; q, \mathcal{I}') \\
&\qquad + \sum_{I \in \mathcal{I}' \setminus \{I_1, I_2\}} \mathbb{P}(\tilde{R}_1 = I; q, \mathcal{I}') \mathbb{P}(\tilde{\pi}^{-1}(I_1) > \tilde{\pi}^{-1}(I_2) | \mathcal{I}' \setminus I) \\
&\quad = \mathbb{P}(\tilde{R}_1 = I_1; q, \mathcal{I}') \\
&\qquad + \sum_{I \in \mathcal{I}' \setminus \{I_1, I_2\}} \mathbb{P}(\tilde{R}_1 = I; q, \mathcal{I}') \mathbb{P}(\tilde{\pi}^{-1}(I_1) > \tilde{\pi}^{-1}(I_2) | \{I_1, I_2\}) \\
&\quad = \frac{\exp(s_\theta(q, I_1)}{\sum_{I' \in \mathcal{I}'} \exp(s_\theta(q, I'))} \\
&\qquad + \sum_{I \in \mathcal{I}' \setminus \{I_1, I_2\}} \frac{\exp(s_\theta(q, I))}{\sum_{I' \in \mathcal{I}'} \exp(s_\theta(q, I))} \frac{\exp(s_\theta(q, I_1))}{\sum_{i \in \{1,2\}} \exp(s_\theta(q, I_i))} \\
&\quad = \frac{\exp(s_\theta(q, I_1)}{\sum_{I' \in \mathcal{I}'} \exp(s_\theta(q, I'))} \\
&\qquad \bullet \ \left(1 + \frac{\sum_{I \in \mathcal{I}' \setminus \{I_1, I_2\}} \exp(s_\theta(q, I))}{\exp(s_\theta(q, I_1)) + \exp(s_\theta(q, I_2))}\right) \\
&\quad = \frac{\exp(s_\theta(q, I_1))}{\exp(s_\theta(q, I_1)) + \exp(s_\theta(q, I_2))} \\
&\quad = \mathbb{P}\left(\tilde{\pi}^{-1}(I_1) > \tilde{\pi}^{-1}(I_2)\ ;\ \{I_1, I_2\}\right)
\end{aligned}$$

This completes our proof. □

B Proof of Theorem 4.4

Given that at time T the algorithm has completed exactly M cycles of the GPER algorithm, one can easily derive that $T_M < T \leq T_{M+1}$ with $T_M = \sum_{m=1}^{M} 2^{m-1} = 2^M - 1$ and that $2T_M + 1 = T_{M+1}$. If we can show that $\mathcal{R}(T) \leq \alpha T^{3/4} + \beta$ at the two boundary points T_M and

T_{M+1}, then the result follows from

$$\begin{aligned}\mathcal{R}(T) &\leq \alpha T_{M+1}^{3/4} + \beta = \alpha(2T_M+1)^{3/4} + \beta \\ &< \alpha(2T+1)^{3/4} + \beta \\ &< \alpha 2^{3/4}(T^{3/4} + (3/4)(1/2)T^{-1/4}) + \beta \\ &< 2\alpha T^{3/4} + 3\alpha/4 + \beta \quad ,\end{aligned}$$

where we used on the second line the fact that because $x^{3/4}$ is concave, $(x+\Delta x)^{3/4} < x^{3/4} + \frac{3}{4}x^{-1/4}\Delta x$ necessarily for $\Delta x > 0$.

It remains to show that at the points T_M the regret is bounded by some $\alpha T_M^{3/4} + \beta$. Since at iteration T_M, the M-th cycle was just completed, regret can be evaluated as follows:

$$\begin{aligned}\mathcal{R}(T_M) &= \sum_{t=1}^{T_M} cost_t(\theta_t) - \min_{\theta\in\Re^n} \sum_{t=1}^{T_M} cost_t(\mathcal{P}_{\Theta_{\phi(t)}}(\theta)) \\ &= \sum_{m=1}^{M}\sum_{t=t_m}^{T_m} cost_t(\theta_t) - \min_{\theta\in\Re^n} cost_t(\mathcal{P}_{\Theta_{\phi(t)}}(\theta)) \quad ,\end{aligned}$$

where $t_m = \sum_{k=1}^{m-1} 2^{k-1} + 1 = 2^{m-1}$ is the iteration at which the m-th cycle started and $cost_t(\mathcal{P}_{\Theta_{\phi(t)}}(\theta))$ stands for $cost(\mathcal{P}_{\Theta_{\phi(t)}}(\theta); q_t, \mathcal{I}_t, I_t, \Phi_t(\cdot))$. We can now use the fact that

$$\min_{\theta\in\Re^n} \sum_{m=1}^{M}\sum_{t=t_m}^{T_m} cost_t(\mathcal{P}_{\Theta_{\phi(t)}}(\theta)) \geq \sum_{m=1}^{M} \min_{\theta_m\in\Theta_m} \sum_{t=t_m}^{T_m} cost_t(\theta_m)$$

to show that $\mathcal{R}(T_M)$ is bounded by the sum of regret cumulated over each cycle.

$$\begin{aligned}\mathcal{R}(T_M) &\leq \sum_{m=1}^{M}\sum_{t=t_m}^{T_m} cost_t(\theta_t) - \min_{\theta_m\in\Theta_m} cost_t(\theta_m) \\ &\leq \sum_{m=1}^{M} \mathcal{R}_{PG}(2^{m-1}; \{q_t, \mathcal{I}_t, I_t, \Phi_t(\cdot)\}_{t=t_m}^{T_m}, \Theta_m) \quad ,\end{aligned}$$

where $\mathcal{R}_{PG}(2^{m-1}; \{q_t, \mathcal{I}_t, I_t, \Phi_t(\cdot)\}_{t=t_m}^{T_m}, \Theta_m)$ refers to the fact that the regret cumulated over the iterations $t_m < t < T_m$ the PGER algorithm and the PG algorithm are equivalent; thus, cumulate the same regret. Since by our assumption $D_m F_m < K 2^{m/4} = 2^{1/4} K (T_m - t_m + 1)^{1/4}$, one can apply Theorem 4.2 to show that:

$$\begin{aligned}\mathcal{R}(T_M) &\leq \sum_{m=1}^{M} 2K 2^{3m/4} = 2^{7/4} K \frac{2^{3M/4}-1}{2^{3/4}-1} \\ &\leq 5K(T_M+1)^{3/4} \leq 5K(T_M^{3/4} + 3/4) \quad .\end{aligned}$$

Thus, $\mathcal{R}(T) \leq 10K(T^{3/4} + 3/4)$ which concludes our proof. □