doi:10.1093/comjnl/bxh000

Adaptive Routing Using Expert Advice

András György[1], György Ottucsák[2]

[1]*Informatics Laboratory, Computer and Automation Research Institute of the Hungarian Academy of Sciences, Lágymányosi u. 11, Budapest, Hungary, H-1111*
[2]*Department of Computer Science and Information Theory, Budapest University of Technology and Economics, Magyar tudósok körútja 2, Budapest, Hungary, H-1117*
Email: {gya, oti}@szit.bme.hu

Machine learning algorithms for combining expert advice in sequential decision problems are considered. The goal of these algorithms is to perform, for any behavior of the system, asymptotically as well as the best expert. We provide a survey of these algorithms and show how they can be used for adaptive routing in different packet switched networks.

Keywords: adaptive routing, combining expert advice, packet switched networks, quality of service, sequential decision problems, shortest path problem

1. INTRODUCTION

Adaptive routing algorithms are of great importance in the maintenance of packet switched communication networks. A sufficiently flexible algorithm can yield increased quality of service, such as reduced packet loss ratio or delay, even in case of link failures or substantially changed traffic scenarios. These algorithms require constant monitoring of network state, and the measured information is combined to update the routing tables. Such combinations can be done in a number of ways, from very simple heuristics to more complicated methods, such as neural networks or reinforcement learning, or sequential decision methods. In this paper we survey the latter.

In sequential decision (prediction) problems in general, a decision maker has to perform a sequence of actions. After each action, the decision maker suffers some loss, depending on the response of the environment. Its goal is to minimize its cumulative loss over a sufficiently long period of time. Adaptive routing can naturally be cast as a sequential decision problem, as for each packet the routing algorithm has to choose a path from source to destination on which the packet is to be sent. The loss corresponding to the decision is the value of the service parameter we wish to minimize, such as the delay on the path or whether the packet is lost due to insufficient buffer size.

In this paper we consider sequential decision problems in the adversarial setting where no probabilistic assumption is made on how the loss of the decision maker is generated, and the goal is to perform well relative to a set of experts for all possible behavior of the environment. More precisely, the aim of the decision maker is to achieve asymptotically the same average loss as the best expert. To solve this problem, the decision maker has access to the decisions of the experts before making his own, and hence can combine them. However, it is impossible to know in advance the performance of the experts. Yet, the experts' advice can be combined such that the average loss of the combined algorithm is asymptotically not larger than that of the best expert over a sufficiently long period of time.

The first theoretical results concerning sequential prediction are due to Blackwell [1] and Hannan [2], but they were rediscovered by the learning community only in the 1990's, see, for example, Vovk [3], Littlestone and Warmuth [4] and Cesa-Bianchi *et al.* [5]. These results show that it is possible to construct algorithms for sequential (online) prediction that predict almost as well as the best expert. The main idea of these algorithms is the same: after observing the past performance of the experts, in each step the decision of a randomly chosen expert is followed such that experts with superior past performance are chosen with higher probability.

In the routing problem, the decisions of the experts can be defined as paths from source to destination, and in the simplest case, one can define one expert for each path. In that case, competing with the best expert results in algorithms that have at least the same asymptotic performance as the *best fixed path*. Stronger results can be obtained by defining more flexible ("more clever") experts, for example by allowing the decisions of the experts to change in time. In this way we can compete with the performance of *time-varying* paths.

In this paper we give an overview of expert algorithms and describe their applications in adaptive routing. For simplicity, we will concentrate on minimizing the average end-to-end delay between two dedicated nodes of the network, but the results can be extended to any other quality of service parameters in a straightforward way. More precisely, we will consider the application of the expert algorithms to find online the minimum weight path between two dedicated nodes of a weighted directed graph. This problem is equivalent to the problem of optimal routing if, at each time instant, the weights of the edges of the graph are matched to the service parameters of the corresponding links to be minimized.

2. SEQUENTIAL DECISION PROBLEMS

The sequential (often referred also as online) decision problem considered in this paper is described as follows. Suppose a decision maker has to make a sequence of actions. At each time instant $t = 1, 2, \ldots, n$, an action $a_t \in \mathcal{A}$ is made, where $\mathcal{A}$ denotes the action space and n is the number of rounds the algorithm is run for. Then, based on the state of the environment $y_t \in \mathcal{Y}$, where $\mathcal{Y}$ is some state space, the decision maker suffers some loss $\ell(a_t, y_t)$ with some bounded loss function $\ell : \mathcal{A} \times \mathcal{Y} \to [0, 1]$. The action at time t may depend on all previous actions $a_1, \ldots, a_{t-1}$, and on all the information available to the decision maker about the past behavior of the environment. This information, for example, may consist of the past environment states $y_1, \ldots, y_{t-1}$; however, the decision maker may not be able to observe the state of the environment. The goal of the decision maker is to minimize the average loss of the algorithm on the long run, that is, to minimize

$$\limsup_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} \ell(a_t, y_t).$$

Since no probabilistic assumption is made on how the sequence $\{y_t\}$ is generated, it is not possible to minimize the cumulative loss $\widehat{L}_n = \sum_{t=1}^{n} \ell(a_t, y_t)$ simultaneously for all $y_1, \ldots, y_n$. Therefore, the performance of the decision maker is evaluated relative to a set of experts, and its goal is to perform asymptotically as well as the best expert. Formally, given N experts, at each time instant t, for every $i = 1, \ldots, N$, expert i chooses its action $f_{i,t} \in \mathcal{A}$ and suffers loss $\ell(f_{i,t}, y_t)$. The decision maker is allowed to make its own decision a_t using the experts' advice $f_{1,t}, \ldots, f_{N,t}$, however, without knowing the experts' loss in advance. When the action space is finite, without loss of generality we may assume that the decision maker always follows the advice of one of the experts, that is, $a_t = f_{I_t,t}$ for some I_t (this can always be achieved by introducing some extra experts). Therefore, we can assume that the decision of the decision maker is to choose an expert I_t and follow its decision $f_{I_t,t}$. Formally, the sequential decision problem is given in Figure 1.

Sequential decision problem using expert advice
Parameters: number N of experts, state space $\mathcal{Y}$, action space $\mathcal{A}$, loss function $\ell : \mathcal{A} \times \mathcal{Y} \to [0, 1]$, n number of rounds.

At time instants $t = 1, \ldots, n$,

(1) each expert forms its action $f_{i,t} \in \mathcal{A}$, $i = 1, \ldots, N$;
(2) the decision maker observes the actions of the experts and chooses an expert $I_t \in \{1, \ldots, N\}$;
(3) the state of the environment $y_t \in \mathcal{Y}$ may or may not be revealed;
(4) the decision maker incurs loss $\ell(f_{I_t,t}, y_t)$ and each expert incurs loss $\ell(f_{i,t}, y_t)$.

FIGURE 1. Sequential decision problem using expert advice.

Throughout the paper, unless otherwise stated explicitly, we assume that the experts are constants (static); that is, $f_{i,t}$ does not depend on t. Then each expert can be considered as an action. For convenience we use the notations $\ell_{i,t}$ instead of $\ell(f_{i,t}, y_t)$ and $\ell_{I_t,t}$ instead of $\ell(f_{I_t,t}, y_t)$. Then the cumulative loss of the decision maker up to time n is

$$\widehat{L}_n = \sum_{t=1}^{n} \ell_{I_t,t},$$

and the cumulative loss of expert i is

$$L_{i,n} = \sum_{t=1}^{n} \ell_{i,t}.$$

The goal of the learning algorithm is to combine the experts decisions such that the *normalized regret*, defined as

$$\frac{1}{n}\left(\widehat{L}_n - \min_{i=1,\ldots,N} L_{i,n}\right),$$

the difference between the average loss of the algorithm and that of the best expert, be universally small for all possible sequences of $\{y_t\}$. It can be shown that under general conditions on the loss function and on the finite action space, excluding such simple situations when, for example, the loss of the experts are the same, no deterministic algorithm can perform well for all possible sequence $\{y_t\}$. This is because for each deterministic algorithm one can construct a "bad" sequence on which the algorithm performs poorly, but the best expert does not. Therefore, in the following we consider randomized algorithms. That is, for each t, I_t is a random variable, as well as the cumulative loss $\widehat{L}_n$. In this case the goal of the algorithm is to perform well with high probability. That is, to ensure

$$\limsup_{n\to\infty} \frac{1}{n}\left(\widehat{L}_n - \min_{i=1,\dots,N} L_{i,n}\right) \le 0$$

with probability 1 for every sequence $\{y_t\}$. Such an algorithm is called Hannan consistent. [6]

As an example to show that deterministic algorithms does not work in general, consider the following example. Assume that we have two actions (experts), with loss sequences $\{1/2, 0, 1, 0, 1, 0, \dots\}$ and $\{0, 1, 0, 1, 0, 1, \dots\}$ and the decision maker's strategy is to always use the decision that has been best so far. This is the so called *follow the leader* strategy. Then except for the first time instant, the first action is chosen at time t if t is odd, and the second action is chosen if t is even, resulting in choosing the worse action for each t. Then the average loss of the algorithm converges to 1, while the loss of both actions are asymptotically 1/2; thus the performance of the algorithm is far from optimal.

However, if the action space is convex (in this case obviously an infinite action space is required), then instead of choosing an action $a_t = i$ (or $I_t = i$) according to a distribution $\{p_{i,t}\}$, we can combine the decisions as $a_t = \sum_{i=1}^N p_{i,t} f_{i,t}$. If the loss function $\ell(\cdot,\cdot)$ is convex in its first argument, then such deterministic algorithms can be applied, see, e.g., Cesa-Bianchi and Lugosi [6]. In routing, for example, this method corresponds to time sharing different paths from source to destination instead of randomly choosing one path.

The performance of any expert algorithm obviously depends on how much information is available to the decision maker about the experts' and its own performance. As we will show in Section 4, if all the information is available about the losses of the past actions, which is called the *full information* case, then there exist algorithms such that the normalized regret $\frac{1}{n}\left(\widehat{L}_n - \min_{i=1,\dots,N} L_{i,n}\right)$ is of order $\sqrt{\ln N/n}$, see, e.g., [3],[4],[5],[7],[8]. However, in certain types of problems it is not possible to obtain full information on the past performance of the experts. For example, in many situations the decision maker has only information on the loss of the chosen action, and no information is available about the loss it would have suffered had it made a different decision. This is called the *multi-armed bandit problem*. Another example is when it is expensive to obtain the losses of the experts, and therefore the decision maker has the option to query this information. In typical cases this corresponds to the response of the environment, also called as outcome or label, from which it is possible to compute the loss of each expert. This type of problem is called *label efficient prediction*. In all of these problems, including the full information case, when the loss of each expert is revealed after the action of the decision maker, there exist algorithms whose average loss is n steps exceeds the average loss of the best of N experts by a negligible term converging to zero as n increases. In the full information case the normalized regret converges to zero as $O(\sqrt{\ln N/n})$, in the multi-armed bandit problem as $O(\sqrt{N \ln N/n})$ [9], and in the label efficient prediction problem with m queries in n rounds as $O(\sqrt{\ln N/m})$ [10]. A recent combination of the multi-armed bandit and label efficient prediction problem, where the loss of the chosen action can be queried m times in n rounds can also be solved with normalized loss of order $\sqrt{N \ln N/m}$ [11]. For a good survey on this topic, the reader is referred to, e.g., the recent book of Cesa-Bianchi and Lugosi [6].

Concerning complexity, for a general class of experts the algorithms typically require $O(nN)$ computations, while the memory requirement is $O(N)$ (one weight has to be updated for each expert at each time instant). While this complexity may be prohibitive for large classes of experts, in many situations, such as the shortest path problem in graphs, the special structure of the experts allows to implement the algorithms with significantly lower complexity, see, e.g., [12],[13],[14], [15],[16],[17],[18],[8],[19],[20].

In this paper we will mainly consider algorithms that are optimized for a fixed time horizon n. That is, the fine tuned parameters of the algorithms depend on n. Such algorithms can be modified easily using standard techniques to perform near optimally for an infinite time horizon [5]: One method is the so called doubling trick, where the time is partitioned into intervals of exponentially increasing length, and a fixed horizon version of the algorithm is run on each interval. However, in practice it is not desirable to reset all accumulated information about the past from time to time, so a better suited technique is to dynamically change the parameters of the algorithms in time.

3. THE ROUTING MODEL AND THE ONLINE SHORTEST PATH PROBLEM

Finding the best route between two dedicated nodes of a network can be considered as the problem of finding the minimum weight path in a weighted directed graph $G = (\mathcal{V}, \mathcal{E})$, representing the network. Each node of the graph corresponds to a node of the network, and the (directed) edges of the graph represent the corresponding links. The weight $\ell_{e,t}$ of each edge $e \in \mathcal{E}$ concerns the service parameter of the corresponding link at time t, in our case, for simplicity, the delay on the link. Suppose we want to send packets from node $u \in \mathcal{V}$ to node $v \in \mathcal{V}$. Let $\mathcal{P}$ denote the set of all directed paths from u to v, and assume that $\mathcal{P}$ is not empty (that is, there is a route from u to v). At any time instant t, a packet is sent from u to v over a (randomly) chosen path $I_t \in \mathcal{P}$. The transmission delay is given by

$$\ell_{I_t,t} = \sum_{e \in I_t} \ell_{e,t}$$

where $e \in I_t$ denotes that edge $e \in \mathcal{E}$ belongs to path I_t, where we assume that for each e and t, $\ell_{e,t} \in [0, 1/K]$ where K denotes the length of the longest path (in the number of hops) from u to v. Similarly, the loss a packet would suffer on any path $i \in \mathcal{P}$ is given by $\ell_{i,t} = \sum_{e \in i} \ell_{e,t}$. Of course, the delay the packet will suffer on any path is not known in advance, and the routing algorithm has to make decisions based on the available information about the past. If performance is compared to static routing, then the goal is to have average delay close to that of the best path matched to the entire sequence of the delays $\{\ell_{e,t} : e \in \mathcal{E}\}_{\{t=1,\dots,n\}}$. That is,

$$\frac{1}{n}\left(\sum_{t=1}^{n} \ell_{I_t,t} - \min_{i \in \mathcal{P}} \sum_{t=1}^{n} \ell_{i,t}\right)$$

has to be as small as possible.

Alternatively, one may want to compete with dynamic routing strategies. More precisely, for a fixed time horizon n, one may want to perform as well as the best time-varying path that is allowed to change the path several times. If s denotes the maximum number of path changes, then such a time-varying routing method is given formally by integers $t_0 = 0 < t_1 < t_2, \dots < t_{s+1} = n$, and $s+1$ paths $i_0, \dots, i_s \in \mathcal{P}$, such that for time instants $t \in (t_k, t_{k+1}]$, $k = 0, \dots, s$, path i_k is used. Let $\mathcal{P}_s$ denote the set of all time-varying paths with s switches. The delay of such an adaptive strategy is given by

$$\sum_{k=0}^{s} \sum_{t=t_k+1}^{t_k} \ell_{i_k,t} = \sum_{k=0}^{s} \sum_{t=t_k+1}^{t_k} \sum_{e \in i_k} \ell_{e,t}.$$

(In general, this type of problems is referred as *tracking the best expert.*) Clearly, both scenarios fall under the framework for sequential decision problems using expert advice: in the static case the experts can be chosen as the paths, and in the dynamic case the experts are the time-varying paths defined above.

Note that the number of experts $N = |\mathcal{P}|$ is typically very large even in the static case (usually exponential in the number of edges), and it is extremely large in the time-varying case even for small graphs as the number of experts $|\mathcal{P}_s|$ is exponential in s. Therefore, direct application of general expert algorithms may be computationally prohibitive, and special algorithms utilizing the graph structure have to be used.

It also has to be analyzed how much information is available to make the routing decision. The delay of the chosen path (or the round-trip delay) is usually available at the source nodes in networks where acknowledgment is sent from the recipient. This scenario can be considered as the shortest path problem in the *multi-armed bandit* setting: information is available on the chosen action, that is, on the chosen path. *Full information* on the network state can be used at each node if the measured network parameters are broadcasted by each node. However, broadcast at each time instant is obviously not desirable, and it may be a good decision to broadcast only a limited number of times. This scenario can be considered as the *label efficient* case. A third type of scenario to be investigated is when the delay of the chosen path is available only on request. This method can be considered as a *combination of the multi-armed bandit and the label efficient* setting, and it is very well suited to the recent cognitive packet network (CPN) framework introduced by Gelenbe *et al.* [21, 22]. In CPN capabilities for routing and flow control are concentrated in packets, and special, so called smart packets, not transporting any data, are used to explore the network (only the chosen path). On the other hand, data packets do not collect information about their paths. Thus, sending a smart packet concerns to query the label of the chosen action.

4. ALGORITHMS

In this section we provide an overview of the most well-known algorithms in different scenarios, and show their specific applications to the shortest path problem. Mostly two types of algorithms are used: The, so called, "follow the perturbed leader"-type algorithms employ the principle (with some additional randomization) that the so far best expert should perform well in the future, too, while weighted average algorithms choose experts randomly such that the ones with better past performance are chosen with higher probability. In the full information case both types of algorithms are given, but in the multi-armed bandit and label efficient settings, we consider only weighted average type algorithms, as for these algorithms better regret bounds are available.

4.1. Full information

In the full information case the decision maker can observe the performance of each expert at each time instant, and hence can utilize all such information in its decision. First we consider the follow the perturbed leader algorithm, then the exponentially weighted average decision method.

4.1.1. Follow the perturbed leader

It was mentioned in Section 2 that the strategy of following the leader (that is, the best expert so far) is not optimal. However, a simple modification suffices to achieve a significantly improved performance, proved by Hannan [2]. The idea is to add small random perturbations to the cumulative losses and then follow

```
Algorithm 1. Follow the perturbed leader (Hannan [2])
Initialization: Fix $R > 0$, and set $L_{i,0} = 0$ for $i = 1, \ldots, N$.
At time instants $t = 1, 2, \ldots$
  (1) Select the random $N$-vector $\mathbf{Z}_t$ with components $Z_{i,t}$, $i = 1, \ldots, N$, uniformly from $[0, R]$;
  (2) Select an expert
        $I_t = \arg\min_{i=1,\ldots,N} (L_{i,t-1} + Z_{i,t})$
      (ties are broken in favor of the smallest index);
  (3) Update the loss of each expert $i$
        $L_{i,t} = L_{i,t-1} + \ell_{i,t}$.
```

FIGURE 2. The follow the perturbed leader algorithm in the full information case.

```
Algorithm 2. Exponentially weighted average decision (Littlestone and Warmuth [4])
Initialization: Fix $\eta > 0$, and set $w_{i,0} = 1$ and $p_{i,1} = 1/N$ for $i = 1, \ldots, N$.
At time instants $t = 1, 2, \ldots$
  (1) Select an expert $I_t \in \{1, \ldots, N\}$ according to the probability distribution
      $\mathbf{p}_t = (p_{1,t}, \ldots, p_{N,t})$;
  (2) Update the weights $w_{i,t} = w_{i,t-1} e^{-\eta \ell_{i,t}}$;
  (3) Calculate the updated probability distribution
        $p_{i,t+1} = \frac{w_{i,t}}{\sum_{j=1}^N w_{j,t}}, \quad i = 1, \ldots, N.$
```

FIGURE 3. Exponentially weighted average decision algorithm in the full information case.

the "perturbed leader" with best "perturbed" past performance.

The following theorem gives an upper bound on the normalized regret of the follow the perturbed leader algorithm given in Figure 2.

Theorem 4.1 (Hannan [2]). *Assume $n, N \geq 1$, $0 < \delta < 1$, and let $R = \sqrt{nN}$. Then, for any sequence $y_1, \ldots, y_n$, the normalized regret of Algorithm 1 can be bounded with probability at least $1 - \delta$ as*

$$\frac{1}{n}\left(\widehat{L}_n - \min_{i=1,\ldots,N} L_{i,n}\right) \leq 2\sqrt{\frac{N}{n}} + \sqrt{\frac{\ln(N/\delta)}{2n}}.$$

The follow the perturbed leader algorithm can be implemented efficiently for the problem of finding the shortest path in a weighted directed graph, which is a special case of the more general geometric expert framework considered by Kalai and Vempala [8]. In their algorithm, one weight is kept for each edge of the graph, showing the cumulative loss of that edge. The perturbed best path is chosen by finding the path with the minimum perturbed weight, where at each time instant, the perturbed weight of each edge e is obtained as the sum of the edge weight plus a random perturbation $Z_{e,t}$ similar to the above. Choosing the optimal path can be done efficiently using the well-known Dijkstra algorithm. This algorithm can be implemented in $O(n|\mathcal{E}|\ln|\mathcal{E}|)$ time, and has $O(\sqrt{|\mathcal{V}||\mathcal{E}|/n})$ expected normalized regret. The performance of the algorithm can be improved slightly if the perturbation is not uniform but it is drawn from a Laplacian distribution. However, in this case the resulting perturbed weights may become negative, which can result in cycles with negative weights in which case the loss can be made an arbitrarily large negative number.

4.1.2. Exponential weighting

In the "weighted average decision"-type algorithms at time instant t an expert i is chosen with probability that increases with the past performance of the expert. That is, $\mathbb{P}\{I_t = i\}$ is proportional to $r(L_{i,t-1})$, where r is a nonincreasing function. The most popular choice of r is $r(x) = e^{-\eta x}$, leading to the exponentially weighted average decision algorithm, given in Figure 3.

As the next theorem shows, in this case the normalized regret of the decision maker is bounded by $O(\sqrt{\ln N/n})$.

Theorem 4.2 (Littlestone and Warmuth [4]). *Let $n, N \geq 1$ and $0 < \delta < 1$. Then, with the choice of $\eta = \sqrt{8 \ln N/n}$, the normalized regret of Algorithm 2 can be bounded for any sequence $y_1, \ldots, y_n$ as*

$$\frac{1}{n}\left(\widehat{L}_n - \min_{i=1,\ldots,N} L_{i,n}\right) \leq \sqrt{\frac{\ln N}{2n}} + \sqrt{\frac{\ln(1/\delta)}{2n}}$$

with probability at least $1 - \delta$.

Remark 1. Algorithm 2 has the disadvantage that the regret bound of Theorem 4.2 does not hold uniformly over sequences of any length n, since the parameter η depends on n. To fix this problem the simplest idea is the *doubling trick* which appears in Cesa-Bianchi *et al.* [5]. The idea is to partition the time into periods of exponentially increasing length. In each period, the algorithm chooses the optimal η for the length of the interval and when the periods end, reset the whole fixed-horizon algorithm, and the new value of η is selected optimally for the next period. This method give an $\sqrt{2}/(\sqrt{2}-1)$ multiplicative factor to the upper bound of the theorem. Another method is that at each time instant t the algorithm chooses an $\eta = \eta_t$ which depends on t. It was proved by Auer *et al.* [23] that setting $\eta_t = \sqrt{8 \ln N/t}$ results in a regret bound that is only twice as much as the original (time dependent) bound.

The above described algorithm can be specialized for graphs with smaller computational complexity [13],[17],[18],[19]. The main idea is to select the edges of the path one by one according to the conditional distributions generated by the exponentially weighted average decision algorithm.

Next, following [19] and [20], we show how to implement the algorithm to compete with paths of a fixed length K. Then it is explained how this algorithm can be extended to compete with all paths, and what simplifications can be made for acyclic graphs.

For any node $w \in \mathcal{V}$, let $\mathcal{P}_w^k$ denote the set of paths of length k from w to v. Let $G_{t-1}(w,k)$ denote the sum of the exponential cumulative losses in the interval $[1, t-1]$ of all paths in $\mathcal{P}_w^k$. That is, if $\mathcal{P}_w^k$ is empty then we define $G_{t-1}(w,k) = 0$, otherwise

$$G_{t-1}(w,k) = \sum_{i \in \mathcal{P}_w^k} e^{-\eta \sum_{e \in i} L_{e,t-1}},$$

where $L_{e,t-1} = \sum_{s=1}^{t-1} \ell_{e,s}$. It can be shown [19] that if I_t is chosen by the exponentially weighted average decision method of Algorithm 2, and $w_{I_t,k}$ denotes the kth node of the path I_t (with $w_{I_t,0} = u$ and $w_{I_t,K} = v$), then

$$\begin{aligned}&\mathbb{P}\left\{w_{I_t,k} = w_k | w_{I_t,0} = w_0, \ldots, w_{I_t,k-1} = w_{k-1}\right\} \\ &\quad = e^{-\eta L_{(w_{k-1},w_k),t-1}} \frac{G_{t-1}(w_k, K-k)}{G_{t-1}(w_{k-1}, K-k+1)}\end{aligned}$$

where (w_{k-1}, w_k) denotes the edge connecting w_{k-1} and w_k, and $G_{t-1}(w_{k-1}, K-k+1) > 0$; if there is no such edge or $G_{t-1}(w_{k-1}, K-k+1) = 0$, then the corresponding conditional probability is formally defined to be 0. The algorithm can be implemented efficiently, as the function $G_t(\cdot, k)$ can be computed recursively for $k = 2, \ldots, K$ as

$$G_{t-1}(w,k) = \sum_{\widehat{w}:(w,\widehat{w}) \in \mathcal{E}} G_{t-1}(\widehat{w}, k-1) e^{-\eta L_{(w,\widehat{w}),t-1}} \quad (1)$$

with

$$G_{t-1}(w,1) = \begin{cases} e^{-\eta L_{(w,\widehat{w}),t-1}} & \text{if } (w,\widehat{w}) \in \mathcal{E}; \\ 0 & \text{otherwise.}\end{cases}$$

It is easy to see that computing the function $G_t(\cdot,\cdot)$ can be done in $O(K|\mathcal{E}|)$ time, hence the whole algorithm can be implemented in $O(nK|\mathcal{E}|)$ time. Competing with the best path of any length can be done by choosing $K = k$ randomly with probabilities proportional to $G_{t-1}(u,k)$ from $\{1, \ldots, |\mathcal{V}|\}$, and then choosing randomly I_t from paths of length K as above. The computational complexity of this algorithm is $O(n|\mathcal{V}||\mathcal{E}|)$, that is typically significantly less than the $O(nN)$ complexity of the original method of Algorithm 2 [20]. For acyclic graphs, there is no need for the second variable of G_t, and a similar algorithm can be performed in $O(n|\mathcal{E}|)$ time, see [13],[17],[18].

4.1.3. Tracking the best expert

So far we considered situations where the goal of the decision maker was to perform as well as the best static expert. However, the performance of static experts is usually rather limited compared to time-varying experts. In this subsection we consider the problem of designing an algorithm that performs as well as the best time-varying expert that can switch the applied experts several times. This problem is usually referred in the literature as the problem of *tracking the best expert*, and was described briefly in the context of adaptive routing in Section 3. Formally, a time-varying expert that is allowed to change the applied static expert s times during a period of length n is given by integers $t_0 = 0 < t_1 < \cdots < t_s < t_{s+1} = n$, experts $i_0, \ldots, i_s \in \{1, \ldots, N\}$ such that at time instants $t \in (t_k, t_{k+1}]$, the time-varying expert follows the static expert i_k. (That is, time is divided into $s+1$ intervals in an arbitrary way, and the time-varying expert behaves as a static expert in each of its intervals.)

To perform asymptotically as well as the best time-varying expert with s switches, one could apply either the exponentially weighted average decision method or the follow the perturbed leader method. However, the number of such time-varying experts is $\sum_{k=0}^{s} \binom{n-1}{k} N(N-1)^k$, which results in prohibitively large complexity for both algorithms; especially, since for any meaningful class, s should also tend to infinity as n increases.

However, Herbster and Warmuth [15] provided an "exponentially weighted average"-type algorithm for this problem that requires the maintenance of one weight for each static expert only, with only slightly increased performance bound. The algorithm given in Figure 4 is a slightly modified version (in step (3)) of the original fixed share algorithm of [15] that appeared in [20].

The performance of Algorithm 3 is bounded by the following theorem.

Theorem 4.3 (Herbster and Warmuth [15] and Vovk [24]). *Let $n, N > 1$, $s \geq 1$, and $0 < \delta < 1$. Then, with the choice of*

$$\alpha = \frac{s}{n-1} \quad \text{and} \quad \eta = \sqrt{8 \ln\left(\frac{N^{s+1}}{\alpha^s (1-\alpha)^{n-s-1}}\right) / n},$$

```
ALGORITHM 3. Tracking the best expert (Herbster and Warmuth [15])

Initialization: Fix $\eta > 0$ and $0 < \alpha < 1$, and let $w_{i,0} = 1$ and $p_{i,1} = 1/N$ for $i = 1, \ldots, N$.
At time instants $t = 1, 2, \ldots$
  (1) Draw $I_t$ randomly according to the distribution $\mathbb{P}\{I_t = i\} = p_{i,t}$;
  (2) After observing $y_t$, for all $i = 1, \ldots, N$, let
          $\widehat{w}_{i,t} = w_{i,t-1} e^{-\eta \ell_{I_t,t}}$;
  (3) For $i = 1, \ldots, N$, set
          $w_{i,t} = \frac{\alpha W_t}{N} + (1-\alpha)\widehat{w}_{i,t}$
      where $W_t = \sum_{i=1}^N \widehat{w}_{i,t}$;
  (4) Calculate the updated probability distribution
          $p_{i,t+1} = \frac{w_{i,t}}{\sum_{j=1}^N w_{j,t}}$.
```

FIGURE 4. Exponential weighting for tracking the best expert.

the normalized regret of Algorithm 3 can be bounded, with probability at least $1-\delta$, *as*

$$\frac{1}{n}\left(\widehat{L}_n - \min_{1 \le t_1 < \ldots < t_s < n} \sum_{k=0}^{s} \min_{i=1,\ldots,N} \sum_{t=t_k+1}^{t_{k+1}} \ell_{i,t}\right) \le \sqrt{\frac{(s+1)\ln N + s \ln \frac{n-1}{s} + s}{2n}} + \sqrt{\frac{\ln(1/\delta)}{2n}}$$

for any sequence $y_1, \ldots, y_n$.

Note that if the number of static experts grows with n as $N = O(n^\gamma)$ for some $\gamma > 0$, then the bound in the theorem becomes $O\left(\sqrt{(s/n)\ln n}\right) = O\left(\sqrt{(s/n)\ln N}\right)$, which is the same (up to a constant factor) as if we competed with the best static expert on a segment of average length.

For other variants of tracking algorithms for general experts, see also [15] and Bousquet and Warmuth [16]. The algorithm in the latter is optimized to the case where each time-varying expert uses only a small set of static experts. Specializing Algorithm 3 to the shortest path problem is much harder than the other "exponential weighting"-type algorithms in this paper because of the mixing step (3). To our knowledge the only such result is [20] providing a tricky implementation of Algorithm 3 with complexity $O(n^2|\mathcal{V}||\mathcal{E}|)$, which is still a factor n larger than desirable. This implementation becomes useful if the number N of static experts grows with n fast enough so that $N/(|\mathcal{V}||\mathcal{E}|) > n$; for example, if the size of the graph (that is, $\mathcal{V}$ and $\mathcal{E}$) grows polynomially in n, and, as usual, N (the number of paths) grows exponentially with $|\mathcal{E}|$.

4.2. Partial information

In this section we overview expert algorithms for situations where the whole information on its own performance and on the past performance of the experts is not available to the decision maker. The algorithms presented here follow the idea of estimating the performance of the experts based on the available information, and then run the exponentially weighted average decision algorithm using the estimated losses. In general, the normalized regret of the algorithms can be bounded by $O\left(\sqrt{N \ln N/(nI)}\right)$ where I is the average number of experts whose performance are revealed to the decision maker at each time instant. We provide algorithms for the label efficient decision and multi-armed bandit problems, as well as for a combination of the preceding two.

4.2.1. Label efficient decisions

In the label efficient decision problem, after choosing its action at time t, the decision maker has the option to query the "label" y_t of the environment. The decision maker is allowed to make on the average m queries in n time instants. To make the algorithm universal, the query times have to be randomized. Therefore, to query a label, the decision maker uses an independent, identically distributed sequence $S_1, S_2, \ldots, S_n$ of Bernoulli random variables with $\mathbb{P}\{S_t = 1\} = \epsilon$ and asks label y_t if $S_t = 1$. If y_t is known, the decision maker can calculate the losses $\ell_{i,t}$ for all $i = 1, \ldots, N$. If $\epsilon = m/n$, then the number of the revealed labels during n rounds is approximately m for large n, and the proportion of labels queried converges to ϵ with probability 1 as n increases.

In order to apply the exponentially weighted average decision method in this case, the losses have to be modified. In Algorithm 4 shown in Figure 5, estimated losses are used instead of the observed losses:

$$\widetilde{\ell}_{i,t} = \begin{cases} \frac{\ell_{i,t}}{\epsilon}, & \text{if } S_t = 1, \\ 0, & \text{otherwise.} \end{cases}$$

Note that $\widetilde{\ell}_{i,t}$ is an unbiased estimate of the true loss $\ell_{i,t}$, as $\mathbb{E}\left[\widetilde{\ell}_{i,t} | S_1^{t-1}, I_1^{t-1}\right] = \ell_{i,t}$.

The following $O(n\sqrt{\ln(4N/\delta)/m})$ upper bound on the normalized regret of Algorithm 4 is due to Cesa-Bianchi *et al.* [10]:

Theorem 4.4 (Cesa-Bianchi *et al.* [10]). *Assume* $n, N \ge 1$ *and* $0 < \delta < 1$. *If Algorithm 4 is run with*

```
ALGORITHM 4. Exponential weighting for label efficient decisions (Cesa-Bianchi et al. [10])

Initialization: Fix $\eta > 0$ and $0 < \epsilon \le 1$, and set $w_{i,0} = 1$ and $p_{i,1} = 1/N$ for $i = 1, \dots, N$.
At time instants $t = 1, 2, \dots$
  (1) Select an action $I_t \in \{1, \dots, N\}$ according to the probability distribution $\mathbf{p}_t = (p_{1,t}, \dots, p_{N,t})$;
  (2) Draw a Bernoulli random variable $S_t$ such that $\mathbb{P}\{S_t = 1\} = \epsilon$;
  (3) if $S_t = 1$ then obtain $\ell_{i,t}$ for all $i$ and compute the estimated loss ($\widetilde{\ell}_{i,t}$)
        $\widetilde{\ell}_{i,t} = \begin{cases} \frac{\ell_{i,t}}{\epsilon}, & \text{if } S_t = 1, \\ 0, & \text{otherwise;} \end{cases}$
  (4) Update the weights $w_{i,t} = w_{i,t-1} e^{-\eta \widetilde{\ell}_{i,t}}$;
  (5) Calculate the updated probability distribution
        $p_{i,t+1} = \frac{w_{i,t}}{\sum_{j=1}^N w_{j,t}} \quad i = 1, \dots, N.$
```

FIGURE 5. Exponentially weighted average decision algorithm in the label efficient setting.

parameters

$$\epsilon = \max\left\{0, \frac{m - \sqrt{2m\ln(4/\delta)}}{n}\right\} \quad \textit{and} \quad \eta = \sqrt{\frac{2\epsilon \ln N}{n}},$$

then the normalized regret of the decision maker can be bounded with probability at least $1 - \delta$ *as*

$$\frac{1}{n}\left(\widehat{L}_n - \min_{i=1,\dots,N} L_{i,n}\right) \le 2\sqrt{\frac{\ln N}{m}} + 6\sqrt{\frac{\ln(4N/\delta)}{m}},$$

where m *is the average number of the revealed labels.*

4.2.2. The multi-armed bandit problem

In the multi-armed bandit problem, the decision maker learns its own loss $\ell_{I_t,t}$ after choosing an action (expert) I_t, but not the value $\ell_{i,t}$ of the other losses for $i \neq I_t$. Thus, the decision maker does not have access to the losses it would have suffered if it had chosen a different action. This means that the decision maker observes only a piece of information at each time instant. The lack of information implies a natural strategy: namely, first the decision maker has to explore the losses of the experts (*exploration phase*) and then it may keep choosing the action with smallest estimated loss for the remaining time (*the exploitation phase*).

In the classical formulation of multi-armed bandit problems (see, e.g., Robbins [25]), it is assumed that, for each action, the losses are randomly and independently drawn with respect to a fixed but unknown distribution. This version is called the *stochastic multi-armed bandit problem* (for a recent efficient solution, see Auer *et al.* [26]). Here we consider a non-stochastic (or worst-case) version of this problem where the sequence $y_1, \dots, y_n$, describing the state of the environment, is generated by a non-stochastic opponent (non-stochastic or adversarial multi-armed bandit problem) [9].

There are three modifications relative to the full information case. First, the modified method uses *gains* instead of losses, defined as

$$g_{i,t} = 1 - \ell_{i,t}.$$

Moreover, in contrast with the label efficient case, we use biased estimates of the gains defined as

$$g'_{i,t} = \begin{cases} \frac{g_{i,t}+\beta}{p_{i,t}}, & \text{if } I_t = i, \\ \frac{\beta}{p_{i,t}}, & \text{otherwise} \end{cases}$$

where the role of parameter β is to control the bias (for $\beta = 0$ we obtain unbiased estimates of the true gains, since then $\mathbb{E}\left[g'_{i,t} \middle| I_1^{t-1}\right] = g_{i,t}$). Finally, a new parameter $0 < \gamma < 1$ is introduced that is used in the exploration phase: for I_{t+1} action i is chosen according to the probability

$$p_{i,t+1} = (1-\gamma)\frac{w_{i,t}}{\sum_{j=1}^N w_{j,t}} + \frac{\gamma}{N}, \qquad i = 1, \dots, N.$$

The role of γ is to ensure that $p_{i,t+1} \ge \gamma/N$ for all $i = 1, \dots, N$. That is, instead of the pure probability distribution generated by exponential weighting, the decision maker uses a mixture of the exponentially weighted average distribution and the uniform distribution, where the latter allows the decision maker to constantly explore all possible actions. The resulting algorithm is given in Figure 6. The algorithm, as well as the following bound on its performance is due to Auer *et al.* [9].

Theorem 4.5 (Auer *et al.* [9]). *For any* $0 < \delta < 1$ *and for any* $n \ge 4N \ln(N/\delta)$*, if Algorithm 5 is run for the multi-armed bandit problem with parameters*

$$\beta = \sqrt{\frac{\ln(N/\delta)}{nN}}, \gamma = \beta N, \text{ and } \eta = \frac{\gamma}{2N},$$

then, with probability at least $1 - \delta$*,*

$$\frac{1}{n}\left(\widehat{L}_n - \min_{i=1,\dots,N} L_{i,n}\right) \le 5\sqrt{N\ln(N/\delta)/n}.$$

Note that the bound of the theorem, unlike to the full information and the label efficient cases, grows

```
ALGORITHM 5. Exponential weighting in the multi-armed bandit setting (Auer et al. [9])

Initialization: Fix $\eta > 0$, $0 < \beta < 1$ and $0 < \gamma < 1$, and set $w_{i,0} = 1$ and $p_{i,1} = 1/N$ for $i = 1, \ldots, N$.
At time instants $t = 1, 2, \ldots$
  (1) Select an action $I_t \in \{1, \ldots, N\}$ according to the probability distribution $\mathbf{p}_t = (p_{1,t}, \ldots, p_{N,t})$;
  (2) Calculate the estimated gains
        $g'_{i,t} = \begin{cases} \frac{g_{i,t}+\beta}{p_{i,t}}, & \text{if } I_t = i, \\ \frac{\beta}{p_{i,t}}, & \text{otherwise;} \end{cases}$
  (3) Update the weights $w_{i,t} = w_{i,t-1} e^{\eta g'_{i,t}}$;
  (4) Calculate the updated probability distribution
        $p_{i,t+1} = (1-\gamma)\frac{w_{i,t}}{\sum_{j=1}^{N} w_{j,t}} + \frac{\gamma}{N}, \quad i = 1, \ldots, N.$
```

FIGURE 6. Exponentially weighted average decision algorithm for the multi-armed bandit problem.

with $\sqrt{N \ln N}$ instead of $\sqrt{\ln N}$. Hence, the bound is not really useful for graphs of even moderate size. To solve this problem and to reduce computational complexity the algorithm has recently been specialized to the online shortest path problem for weighted acyclic graphs [27]. The method of [27] is similar in spirit to the full information case; that is, the estimated gains are calculated and stored for the edges instead of the paths. The resulting upper bound is also improved: the normalized regret is at most $O(\sqrt{K^2|\mathcal{E}|\ln(N/\delta)/n})$, where K is the maximum path length from u to v. That is, the very large $\sqrt{N}$ factor is replaced with the much smaller factor $K\sqrt{|\mathcal{E}|}$. This is achievable because when the decision maker learns the loss of each edge of a path, then at the same time it learns information on the losses of other paths having common edges with the chosen path.

We note here that "follow the perturbed leader"-type algorithms can also be applied to solve the online shortest path problem in the multi-armed bandit setting. Indeed, Awerbuch and Kleinberg [28] and McMahan and Blum [29] provided such algorithms. However, the obtained bounds do not decrease to zero at a desired $O(1/\sqrt{n})$ rate.

4.2.3. *A combination of the label efficient decision and the multi-armed bandit problems*

In this subsection we introduce a recent combination of the label efficient and the multi-armed bandit problems [11]. This combination was motivated by the routing problem in cognitive packet networks described in Section 3. In this combined problem, the decision maker learns its own loss only if it chooses to query it (which is allowed only for a limited number of times), and it cannot obtain information on the performance of any other action.

This problem is solved with a combination of the algorithms for the label efficient decision problem and for the multi-armed bandit problem. In the combined method, shown in Figure 7, at each time instant t, the algorithm queries the result of its action with probability ϵ (just as in the label efficient case), and similarly to the multi-armed bandit case, it computes biased estimates $g'_{i,t}$ of the true gains $g_{i,t}$ as

$$g'_{i,t} = \begin{cases} \frac{g_{i,t}+\beta}{p_{i,t}\epsilon}, & \text{if } I_t = i \text{ and } S_t = 1; \\ \frac{\beta}{p_{i,t}\epsilon}, & \text{if } I_t \neq i \text{ and } S_t = 1; \\ 0 & \text{otherwise.} \end{cases}$$

Again, $g'_{i,t}$ is an unbiased estimate of $g_{i,t}$ for $\beta = 0$, since then $\mathbb{E}\big[g'_{i,t}|S_1^{t-1}, I_1^{t-1}\big] = g_{i,t}$.

The performance of Algorithm 6 is analyzed in the next theorem of [11], which is, in fact, a joint extension of Theorem 4.4 and Theorem 4.5.

THEOREM 4.6 (OTTUCSÁK AND GYÖRGY [11]). *Assume that $0 < \delta < 1$, $0 < \epsilon \leq 1$, and $n \geq 4N\ln(2N/\delta)/\epsilon$. Then for parameters $\beta = \sqrt{\frac{\ln(2N/\delta)}{nN\epsilon}}$, $\gamma = \beta N$, and $\eta = \frac{\gamma\epsilon}{2N}$, the normalized regret of Algorithm 6 can be bounded with probability at least $1-\delta$ as*

$$\frac{1}{n}\left(\widehat{L}_n - \min_{i=1,\ldots,N} L_{i,n}\right) \leq 12\sqrt{\frac{N\ln(2N/\delta)}{n\epsilon}} + \frac{\ln(2/\delta)}{n\epsilon}.$$

Algorithm 6 can also be modified to suit to the online shortest path problem, with lower computational complexity and improved performance bound: similarly to the multi-armed bandit setting, in the regret bound of the new algorithm the constant N is replaced with $K^2|\mathcal{E}|$ [30].

5. CONCLUSION

In this paper we gave an overview of a special class of machine learning algorithms for sequential decision problems. These algorithms provide efficient methods to combine expert advice in an optimal way in the sense that the algorithms have asymptotically the same performance as the best expert. The proposed methods are universal as they do not require any statistical description of the system, and work for any behavior of the system.

Algorithm 6. **Exponential weighting for the label efficient multi-armed bandit problem (Ottucsák and György [11])**

Initialization: Fix $\eta > 0$, $0 < \beta < 1$, $0 < \gamma < 1$ and $0 < \epsilon \leq 1$, and set $w_{i,0} = 1$ and $p_{i,1} = 1/N$ for $i = 1, \ldots, N$.
At time instants $t = 1, 2, \ldots$

(1) Select an action $I_t \in \{1, \ldots, N\}$ according to the probability distribution $\mathbf{p}_t = (p_{1,t}, \ldots, p_{N,t})$.
(2) Draw a Bernoulli random variable S_t such that $\mathbb{P}\{S_t = 1\} = \epsilon$.
(3) If $S_t = 1$ then obtain $g_{I_t,t}$ and compute the estimated gains

$$g'_{i,t} = \begin{cases} \frac{g_{i,t}+\beta}{p_{i,t}\epsilon}, & \text{if } I_t = i,\ S_t = 1, \\ \frac{\beta}{p_{i,t}\epsilon}, & \text{if } I_t \neq i,\ S_t = 1, \\ 0 & \text{otherwise.} \end{cases}$$

(4) Update the weights $w_{i,t} = w_{i,t-1}e^{\eta g'_{i,t}}$.
(5) Calculate the updated probability distribution

$$p_{i,t+1} = (1-\gamma)\frac{w_{i,t}}{\sum_{j=1}^{N} w_{j,t}} + \frac{\gamma}{N}, \qquad i = 1, \ldots, N.$$

FIGURE 7. Combination of the label efficient and the multi-armed bandit exponentially weighted average forecaster.

We showed how these sequential decision algorithms can be applied in adaptive routing for packet switched networks to ensure increased quality of service. Different scenarios were investigated on how much information is available to the routing algorithm. In practice, applicability of these algorithms depend on how much prior (statistical) information is available about the traffic over the network. Expert algorithms are the most useful when the goal is to design routing algorithms that perform well even for rare, unexpected behaviors of the network.

ACKNOWLEDGMENT

The authors would like to thank László Györfi for useful discussions. This work was supported in part by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences, and by the Mobile Innovation Center of Hungary.

REFERENCES

[1] Blackwell, D. (1956) An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, **6**, 1–8.

[2] Hannan, J. (1957) Approximation to bayes risk in repeated plays. In Dresher, M., Tucker, A., and Wolfe, P. (eds.), *Contributions to the Theory of Games*, pp. 97–139. Princeton University Press.

[3] Vovk, V. (1990) Aggregating strategies. *Proceedings of the Third Annual Workshop on Computational Learning Theory*, Rochester, NY, Aug., pp. 372–383. Morgan Kaufmann.

[4] Littlestone, N. and Warmuth, M. K. (1994) The weighted majority algorithm. *Information and Computation*, **108**, 212–261.

[5] Cesa-Bianchi, N., Freund, Y., Helmbold, D. P., Haussler, D., Schapire, R., and Warmuth, M. K. (1997) How to use expert advice. *Journal of the ACM*, **44**, 427–485.

[6] Cesa-Bianchi, N. and Lugosi, G. (2006) *Prediction, Learning, and Games.* Cambridge University Press, to appear, Cambridge.

[7] Vovk, V. (1998) A game of prediction with expert advice. *Journal of Computer and System Sciences*, **56**, 153–173.

[8] Kalai, A. and Vempala, S. (2003) Efficient algorithms for the online decision problem. In Schölkopf, B. and Warmuth, M. (eds.), *Proceedings of the 16th Annual Conference on Learning Theory and the 7th Kernel Workshop, COLT-Kernel 2003*, New York, USA, Aug., pp. 26–40. Springer.

[9] Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (1995) Gambling in a rigged casino: the adversial multi-armed bandit problem. *Proceedings of the 36th Annual Symposium on Foundations of Computer Science, FOCS 1995*, Washington, DC, USA, Oct., pp. 322–331. IEEE Computer Society Press, Los Alamitos, CA.

[10] Cesa-Bianchi, N., Lugosi, G., and Stoltz, G. (2005) Minimizing regret with label efficient prediction. *IEEE Trans. Inform. Theory*, **IT-51**, 2152–2162.

[11] Ottucsák, Gy. and György, A. (2005). A combination of the label efficient and the multi-armed bandit problems in adversarial setting. Preprint.

[12] Schapire, R. E. and Helmbold, D. P. (1997) Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, **27**, 51–68.

[13] Mohri, M. (1998) General algebraic frameworks and algorithms for shortest distance problems. Technical Report 981219-10TM. AT&T Labs Research.

[14] Auer, P. and Warmuth, M. K. (1998) Tracking the best disjunction. *Machine Learning*, **32**, 127–150.

[15] Herbster, M. and Warmuth, M. K. (1998) Tracking the best expert. *Machine Learning*, **32**, 151–178.

[16] Bousquet, O. and Warmuth, M. K. (2002) Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, **3**, 363–396.

[17] Takimoto, E. and Warmuth, M. K. (2002) Path kernels and multiplicative updates. In Kivinen, J. and Sloan, R. H. (eds.), *Proceedings of the 15th Annual Conference on Computational Learning Theory, COLT*

2002, Berlin–Heidelberg, Jul. LNAI 2375, pp. 74–89. Springer-Verlag.

[18] Takimoto, E. and Warmuth, M. K. (2003) Path kernels and multiplicative updates. *Journal of Machine Learning Research*, **4**, 773–818.

[19] György, A., Linder, T., and Lugosi, G. (2004) Efficient algorithms and minimax bounds for zero-delay lossy source coding. *IEEE Transactions on Signal Processing*, **52**, 2337–2347.

[20] György, A., Linder, T., and Lugosi, G. (2005) Tracking the best of many experts. *Proceedings of the 18th Annual Conference on Learning Theory, COLT 2005*, Bertinoro, Italy, Jun., pp. 204–216. Springer.

[21] Gelenbe, E., Gellman, M., Lent, R., Liu, P., and Su, P. (2004) Autonomous smart routing for network QoS. *Proceedings of First International Conference on Autonomic Computing*, New York, May, pp. 232–239. IEEE Computer Society.

[22] Gelenbe, E., Lent, R., and Xhu, Z. (2001) Measurement and performance of a cognitive packet network. *Journal of Computer Networks*, **37**, 691–701.

[23] Auer, P., Cesa-Bianchi, N., and Gentile, C. (2002) Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, **64**, 48–75. A preliminary version has appeared in *Proc. 13th Ann. Conf. Computational Learning Theory.*

[24] Vovk, V. (1999) Derandomizing stochastic prediction strategies. *Machine Learning*, **35**, 247–282.

[25] Robbins, H. (1952) Some aspects of the sequential design of experiments. *Bullettin of the American Mathematical Society*, **55**, 527–535.

[26] Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002) Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, **47**, 235–256.

[27] György, A., Linder, T., and Lugosi, G. The shortest path problem in the bandit setting. In preparation.

[28] Awerbuch, B. and Kleinberg, R. D. (2004) Adaptive routing with end-to-end feedback: distributed learning and geometric approaches. *Proceedings of the 36th Annual ACM Symposium on the Theory of Computing, STOC 2004*, Chicago, IL, USA, Jun., pp. 45–53. ACM Press.

[29] McMahan, H. B. and Blum, A. (2004) Online geometric optimization in the bandit setting against an adaptive adversary. *Proceedings of the 17th Annual Conference on Learning Theory, COLT 2004*, Banff, Canada, Jul., pp. 109–123. Springer.

[30] Ottucsák, Gy. (2005). Online shortest path problem for a combination of the label efficient and the multi-armed bandit settings. Preprint.