Competitive Algorithms for On-line Set Cover or How to Beat Murphy's Law

Baruch Awerbuch[*] *Yossi Azar*[†] *Amos Fiat*[‡]

Abstract

This paper considers an on-line optimization version of the *set cover* problem. We present a optimally competitive on-line randomized algorithm which is $O(\log n \log m)$ competitive where n is the maximum number of sets and m is maximum the number of elements. Moreover, we provide a matching lower bound for the problem. We also give several applications of the results.

Randomization is crucial for our result since a deterministic algorithm may cover only one element with each accepted set and thus, cannot achieve any non-trivial bound.

1 Introduction

1.1 The Problem

In this paper we consider an on-line version of the following variant of the *set cover* problem, parameterized by k: Given a family of sets $F = \{S_1, S_2, \ldots, S_n\}$, where $S_i \subset \{v_1, v_2, \ldots, v_m\}$, for all i, choose some subset $F' \subset F$, where F' is of size k, such that the size of the union of all $S \in F'$ is maximized.

This paper introduces and optimally solves an on-line version of this problem: Base set elements arrive on-line. When a base element arrives, it reveals to what sets it belongs to. At any point in time we can decide to accept a set, and are limited to accepting no more than k such sets. The algorithm's profit is the number of elements contained in sets that were accepted at or before the time of the element arrival. That is, if we have not yet accepted any of the sets containing an element e and don't accept any of these sets at the

[*]Johns Hopkins University and Lab. for Computer Science, MIT. Supported by Air Force Contract TNDGAFOSR-86-0078, ARO contract DAAL03-86-K-0171, NSF contract 9114440-CCR, DARPA contract N00014-J-92-1799, and a special grant from IBM. E-Mail: baruch@theory.lcs.mit.edu.

[†]Department of Computer Science, Tel Aviv University. E-Mail: azar@math.tau.ac.il. Research supported in part by Allon Fellowship and by the Israel Science Foundation administered by the Israel Academy of Sciences.

[‡]Department of Computer Science, Tel-Aviv University, Israel. E-Mail: fiat@math.tau.ac.il. Research supported in part by the Israel Science Foundation administered by the Israel Academy of Sciences.

arrival time of e, in which e reveals the sets that contains it, then the base element e will not be counted in our profit, even if we do accept some of these sets later.

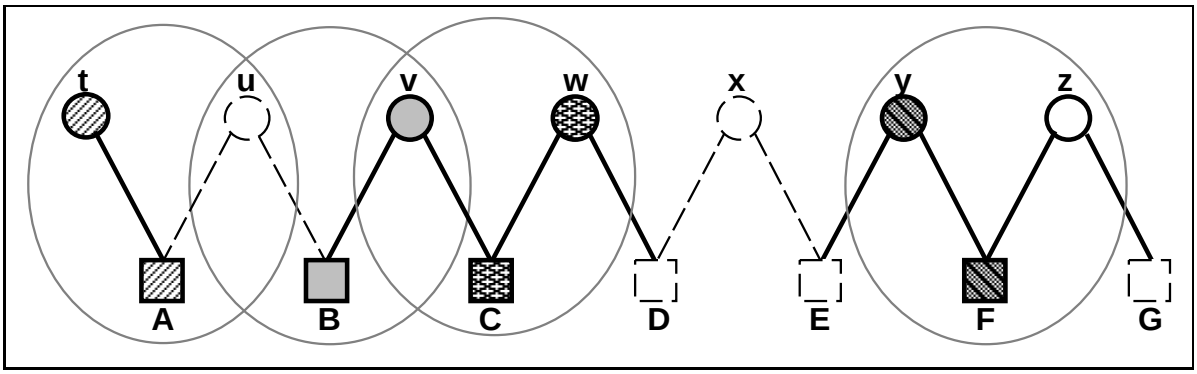

Figure 1: Example of on-line set cover. The elements are shown on top and the sets are shown on the bottom of the bipartite graph, whose edges represent inclusion of elements in the sets. Solid circles are used for elements that have already arrived. Solid squares are used for sets already accepted. Same background color for an element and a set indicates that the element's is counted for the profit as result of purchasing the set prior or at the time of the element's arrival. In the above sequence, t arrived first and A was accepted, v arrived and B was accepted, w arrived and C was accepted. Then, z arrived and nothing was accepted (neither F nor G) and y arrived and F was accepted. In the future, u and x will arrive and no additional sets will be accepted.

The main result in this paper is a strongly competitive randomized solution to the problem, accompanied by matching lower bounds. Our algorithm achieves $O(\log n \log m)$ competitive ratio, where m is the maximum number of sets and n is the maximum number of elements in the system. We will show that randomization is crucial for our result. A deterministic algorithm can receive at most profit of 1 for each accepted set and thus cannot achieve any non-trivial bound.

1.2 Related work

Certainly, on-line optimization variants of combinatorial problems have been subject to competitive analysis [ST85a, KMRS88] in many recent papers: *on-line matching* [KVV90], *partition* [FKT89], *on-line steiner tree* and generalizations [IW91, CV92, ABF93, WY93], and *on-line graph coloring* [Vish90, Irani90, HS92].

A restricted version of our problem that we call the on-line *disjoint* set cover problem, where every element belongs to exactly one set, was considered by Aggarwal, Garay and Herzberg in [AGH94], where it was referred to as "Video-on-Demand" scheduling. Surprisingly, already this problem is non-trivial. The essence of the problem in [AGH94] is that different customers issue requests for different movies to a video server with limited capacity k. A customer demand for a specific movie is either *accepted* or *rejected*, the goal being to maximize the number of satisfied customers subject to the restricted channel capacity, that does not allow to simultaneously show more than k movies. The on-line algorithms in [AGH94] achieve linear ($O(m)$) competitive ratio; the semi-offline algorithms with look-ahead (where the decision can be postponed to the time when future demand is known)

achieve logarithmic ratio. In this case, movies are the sets, and customers are the base elements. The more general (non-disjoint) set cover problem which we deal with corresponds to a setting in which every customer discloses a list of alternative movie titles that he/she wants to watch.

The disjoint set cover problem can also be viewed as the problem of making irrevocable ("no way out") investment decisions: the sets are various companies, and the base elements are the dividends paid by these companies. The algorithm invests some fixed amount of money in one or more of these companies so as to maximize the total dividends. Alternatively, one can think of sets are patents, and elements being opportunities to license these patents. The investment decision is to choose what patents one should purchase.

Previous work on the competitive analysis of financial problems [EFKT92, EK93, CEL93] focused on trading problems, where algorithms were allowed to make partial investments, to retract from previous decisions (at some cost), or were based on some statistical knowledge of the input.

In contrast, on-line set cover captures situations where investment decisions are indivisible and irrevocable: once an investor has decided to invest in building a new factory, he/she must hope that there will be high demand for the product produced by the factory, in which case the dividends paid to investors will be high. In our model, investment decisions do not have to take place immediately; investors can postpone the decision until he/she get convinced that the company is doing well and indeed paying lofty dividends. Still there is risk involved since future and past are not necessarily related, and, by Murphy's law, once the investor decides to invest in a company, demand for for its product drops down to zero, in which case the investor is stuck with the factory building and a warehouse full of merchandise that nobody wants. Hence, with limited $ resources, the investor has to decide what investments to make and at what stage along the curve of increasing demand.

The more general (non-disjoint) set cover problem captures scenarios where there may be complex relationships between between one company's growth and another's, in case they are producing competitive, and, from customer's view, mutually exclusive lines of products, e.g., one product being answering machines and another being a cord-less phone with answering machine. From the investor point of view, the marginal revenues of investing in a factory producing just answering machines may be too low *after* purchase of a factory producing answering machines with a cordless phone.

To capture this problem, we view companies as sets of products, and arriving customers as base elements. Upon their arrival, customers disclose mutually exclusive lists of products they would like to purchase. If, by that time, the investor has already has purchased a company producing any one of these products, the investor makes a profit. Otherwise, the customer (and the money) are lost, since the client will not wait for us. It is the true demand patterns seen in the past that determine what products we will eventually invest in.

1.3 Our results

It should be perfectly obvious that the naturally applicable version of Murphy's law states that whenever we invest in a certain product line, It immediately becomes technically obsolete and future demand drops to zero. Thus, non-oblivious adversaries ensure that we get at most one client per product, leading to a competitive ratio of $\Theta(m/k)$.

The contributions of this papers are matching lower and upper bounds for randomized algorithms with lower bounds applying even to the restricted case of disjoint set cover cover problem, and even for $k = 1$. Specifically, we show

Theorem 1.1 *For the on-line set cover problem:*

- *The randomized algorithm MRT in Section 2 achieves an* $O(\log n \log m)$ *competitive ratio.*
- *The competitive ratio is* $\Omega(\log n \log m)$.

Our result is obtained by using a certain "multiple random threshold" technique, which allows us to "catch" the market in the "middle" of the increasing demand curve. This multiple random threshold technique may be of value in other on-line decision-making settings.

We note that we can generalize our results in a number of ways, sets can have non uniform costs and elements may provide non uniform benefits, elements may reappear several times and may disclose to what sets they belong to over time. For simplicity and clarity, we discuss the simpler case here.

The rest of this extended abstract proceeds as follows. In Section 2, we present the algorithm and its analysis, and in Section 3 we present the lower bound.

2 Algorithms and their analysis

2.1 Basic definitions

Consider an algorithm, on-line or off-line, that solves our problem, and accepts sets in some order. As a function of the time, t, we define the *marginal revenue* of a set S that has not yet been accepted to be the number of elements in S that are not in any of the other sets accepted by the algorithm up to time t.

For an off-line algorithm, the marginal revenue is computed based on the real sets. Consider an off-line algorithm that accepts k sets in some order, its profit is the sum of the marginal revenues of these sets defined by that order.

For an on-line algorithm, the marginal revenue at time t is computed based on the elements known to belong to the sets at time t. The marginal revenue of a set S increases by one when an element arrives that belongs to S and does not belong to any set already accepted. The marginal benefit of a set S may decrease when some set that has elements in common with S is accepted.

For sets that have already been accepted, we define their marginal revenue to be the number of elements in S that are not in any of the sets accepted before S. Note that once a set S has been accepted the marginal revenue cannot decrease any more. For an on-line algorithm, the marginal revenue of a set S increases by one with the arrival of an element, contained in S and not contained in any of the sets accepted before S.

As a function of time t we define the marginal profit of a set S, for sets S that were already accepted, to be the marginal revenue of S at time t excluding all elements that arrived strictly before S was accepted. Clearly, the marginal profit of S cannot decrease. For an on-line algorithm, the marginal profit of a set S increases by one when the marginal revenue of S increases by one.

The total number of elements for which the on-line algorithm gets profit is the sum of the marginal profits of the sets accepted, after the last element in the sequence arrives.

2.2 Off-line Approximation Algorithms

We will uses the following notations:

- k – the number of sets that can be chosen.
- n – the maximum number of sets.
- m – the maximum number of the base elements.
- l – the total number of base elements covered by the optimal off-line algorithm.
- l' – the largest power of two which is not larger than l.
- $q = l'/(2k)$.

We also note that we can avoid the requirement of knowing n and m in advance, at a cost of a small factor in the competitive ratio, by using techniques of [LT94].

We first consider the following type of off-line approximation algorithms. Here all sets and their elements are known in advance. Assume that we are given the value q. Order the sets in an arbitrary order and consider them one by one until k sets have been accepted according to the criteria below or there are no more sets left.

- If the marginal revenue of a set is at least q then accept it.

- If the marginal revenue of a set is at most $q/2$ then reject it.
- Otherwise either accept or reject the set.

We first prove the following

Theorem 2.1 *Any algorithm of the type above approximates the optimum value l to within a constant factor.*

Proof: If the algorithm accepts k sets then they cover at least $kt/2 = l'/4 \geq l/8$ elements and we are done. If the algorithm accepts less than k sets than we show that at least $l/2$ elements are covered.

Assume that the algorithm chosen sets cover less than $l/2$ elements. Consider the sets chosen by the optimal algorithm and not by the approximation algorithm. The union of these sets contains at least $l - l/2 = l/2$ elements which are not covered by the approximation algorithm. Thus, at least one of them had marginal revenue $(l/2)/k \geq q$ at the time that it was considered and thus would not have been rejected by the approximation algorithm. This is a contradiction. ■

2.3 The On-line Algorithm

The algorithm, denoted MRT, (Multiple Random Thresholds), is defined as follows:

1. Choose value uniformly at random among the powers of 2 in the range 2^0 to $2^{\log m}$ and compute q as if the value is l'.
2. For each set j choose a random threshold x_j among the powers of $a = (1 - 1/(4 \log n))$. Specifically, $x_j = a^r$ for $1 \leq r < 2 \log n$ with probability 2^{-r} and for $r = 2 \log n$ with probability $2/n^2$.
3. The j'th set is accepted by the algorithm when its marginal revenue exceeds $q \cdot x_j$.

Clearly the probabilities for choosing r defines a probability distribution and $1/2 < x_j < 1$ for all admissible j.

When a new element arrives the marginal revenue of several sets may reach their threshold. The algorithm accepts an arbitrary set among those that reached their threshold. As a result the marginal revenue of other sets may drop below their threshold. We state the main theorem.

Theorem 2.2 *The MRT algorithm above is $O(\log n \log m)$ competitive.*

2.4 The Analysis

We fixed the sequences of requests (arrival of elements). The vector x of thresholds defines the sets accepted by the algorithm and the order in which they were accepted.

- $I_{(x,j)}$ is an indicator variable. It is 1 if the j'th set is accepted for the threshold vector x and it is 0 otherwise.
- $x_{\hat{j}}$ the vector x without the j'th coordinate.
- $< y, x_j >$ expanding the vector y by inserting x_j as the j'th coordinate. Clearly $< x_{\hat{j}}, x_j >= x$.

A given vector x induces an order of acceptance for the on-line algorithm. We associate with the on-line algorithm an off-line approximation algorithm APP(x) which receives the same sets that the on-line algorithm accepted in the order that the on-line algorithm accepted them, other sets appear later in some arbitrary order. Given l', as the on-line algorithm random threshold $1/2 < x_j < 1$, and as the final marginal revenue is at least as large as the marginal revenue considered by the on-line algorithm when it decides to accept or reject a set, it follows that there is an approximation algorithm of the type used in lemma 2.1 that accepts exactly the same sets as the on-line algorithm accepts, let APP be that algorithm. As with any off-line algorithm, its profit is the sum of the the marginal revenues for the sets accepted.

From now on, we consider the on-line algorithm definitions of marginal revenue and marginal profit at the time of the end of the sequence. Thus, the marginal revenue definitions of the on-line algorithm and of APP are identical, the value of the on-line algorithm is the sum of the marginal profits over the sets accepted.

For the order induced by x and for $1 \leq j \leq n$ denote by $b_j(x)$ the marginal revenue of the j'th set and denote by $p_j(x)$ the marginal profit of a set S. Also denote by $B_{app}(x)$ the value of the approximation algorithm APP(x) and by $B_{on}(x)$ the value of the on-line algorithm MRT(x). As explained above,

$$B_{app}(x) = \sum_j I_{(x,j)} b_j(x)$$

and

$$B_{on}(x) = \sum_j I_{(x,j)} p_j(x).$$

Given that the value l' is correct, and noting that $1/2 < x_j < 1$, it follows from theorem 2.1 implies that for all x in the support $B_{app}(x) = \Omega(l)$ and hence

$$E_x(B_{app}) = \Omega(l) \ .$$

Moreover, the value of the on-line algorithm for the vector x is the sum of the marginal profits of the accepted sets which is

$$\begin{aligned} B_{on}(x) &= \sum_j I_{(x,j)} p_j(x) \\ &= \sum_j I_{(x,j)} \lfloor (b_j(x) - q \cdot x_j + 1) \rfloor \\ &\geq \sum_j I_{(x,j)} (b_j(x) - q \cdot x_j) \end{aligned}$$

Next we make the observations that decreasing the threshold of a set cannot decrease its marginal revenue and cannot postpone the event of set acceptance.

Lemma 2.1 *$I_{(x,j)}$ and $b_j(x)$ are monotone non-increasing functions of x_j which is the j'th coordinate of x.*

Proof: The Lemma follows from the fact that decreasing the value of x_j may cause only to reduce the family of earlier accepted set of the j'th set. That may only increase the marginal revenue of the j'th set and get the set to be accepted earlier. ■

For a fixed $x_{\hat{j}}$ we identify the threshold of which a set is accepted as follows. Let r_j the minimum value of r, $1 \leq r \leq 2 \log n$, such that

$$b_j(< x_{\hat{j}}, a^r >) \geq q \cdot a^r.$$

If such r does not exists define $r_j = 2 \log n + 1$. Note that the left hand side is monotone non-decreasing function of r where the right hand side is monotone decreasing function of r. Denote $< x_{\hat{j}}, a^r >$ by $x(r)$. The following lemma claims that as long as the exact threshold of a set is chosen then either the set is not chosen or its marginal profit is comparable to the marginal revenue.

Lemma 2.2 *In the above notations $p_j(x(r)) \geq b_j(x(r))/(4 \log n)$ for $r > r_j$. On the other hand $I_{(x(r),j)} = 0$ for $r < r_j$.*

Proof: If $1 \leq r < r_j$ then $b_j(x(r)) < q \cdot a^r$ and thus it will not be accepted by the on-line algorithm and the approximation algorithm and hence $I_{(x(r),j)} = 0$. If $r > r_j$ then

$$b_j(< x_{\hat{j}}, a^r >) \geq b_j(< x_{\hat{j}}, a^{r_j} >) \geq q \cdot a^{r_j} \geq q \cdot a^r a^{-1}$$

Hence

$$b_j(x(r))(1 - 1/(4 \log n)) \geq q \cdot a^r$$

or

$$p_j(x(r)) \geq b_j(x(r)) - q \cdot a^r \geq b_j(x(r))/(4 \log n).$$

■

We are ready to prove the main theorem. Denote $x' =< x_{\hat{j}}, a^{2\log n} >$. Let P_x be the probability of choosing the threshold vector x. It follows from the probability distribution defined by the algorithm for each coordinate and from the independence between different coordinates that for $1 \leq r \leq 2\log n - 1$

$$P_{<x_{\hat{j}},a^r>} = P_{<x_{\hat{j}}>}P_{a^r} \leq 2P_{<x_{\hat{j}}>}P_{a^{r+1}} = 2P_{<x_{\hat{j}},a^{r+1}>}.$$

The above, lemma 2.2, and lemma 2.1 imply that

$$\begin{aligned}
\sum_{r=1}^{2\log n} P_{x(r)}I_{(x(r),j)}b_j(x(r)) &= \sum_{r \geq r_j} P_{x(r)}b_j(x(r)) \\
&= \sum_{r-1 \geq r_j} P_{x(r-1)}b_j(x(r-1)) \\
&\leq \sum_{r > r_j} 2P_{x(r)}b_j(x(r)) + P_{x'}b_j(x').
\end{aligned}$$

Apply lemma 2.2 and the definition of $P_{x'}$ to bound the above by

$$\begin{aligned}
\sum_r P_{x(r)}I_{(x(r),j)}b_j(x(r)) &\leq (4\log n)\cdot \sum_{r>r_j} 2P_{x(r)}I_{(x(r),j)}p_j(x(r)) + \frac{2}{n^2}P_{x_{\hat{j}}}b_j(x') \\
&\leq 8\log n \sum_r P_{x(r)}I_{(x(r),j)}p_j(x(r)) + \frac{2}{n^2}P_{x_{\hat{j}}}b_j(x')
\end{aligned}$$

Now we sum the above inequality over all j and $x_{\hat{j}}$ and conclude

$$\begin{aligned}
E_x(B_{app}) &= \sum_j \sum_{x_{\hat{j}}} \sum_r P_{x(r)}I_{(x,j)}b_j(x(r)) \\
&\leq 8\log n \sum_j \sum_{x_{\hat{j}}} \sum_r P_x(r)I_{(x(r),j)}p_j(x(r)) + \frac{2}{n^2}\sum_j \sum_{x_{\hat{j}}} P_{x_{\hat{j}}}b_j(x') \\
&\leq 8\log n E_x(B_{on}(x)) + \frac{2}{n^2}nl
\end{aligned}$$

But since given the value l' and $E_x(B_{app}) = \Omega(l)$ we conclude that

$$E_x(B_{on}(x)) = \Omega(E_x(B_{app}/\log n)) = \Omega(l/\log n).$$

To complete the proof of the main theorem we note that we choose the correct value for l' with probability $\Omega(1/\log m)$. Thus the competitive ratio is $O(\log n \log m)$ as needed.

3 The lower bounds

The lower bounds are proved even for the special case of disjoint sets That is each element belongs to precisely one set. i.e, the sets are disjoint. We first observe that $m/(2k)$ is an easy lower bound for deterministic algorithms (which is obviously tight). First, elements that belong to the first set arrive. Once the set is taken elements of the second set arrive and so forth. For each such set the on-line get 1 and thus at most k for all sets. If after k such sets at least $m/2$ elements were used so the off-line accepts all these set in advance and receives a profit of $m/2$. Otherwise $m/2$ elements arrive for the $k+1$ set. The off-line accepts this set with profit $m/2$ where the on-line remains with the k as before since it cannot accept any more sets. In any case the off-line got a profit of at least $m/2$ where the on-line received at most k. That implies the lower bound.

The randomized lower bound is more complicated. We prove it for $k = 1$. It can easily be extended to the case $k > 1$. The competitive ratio of a randomized algorithm is defined as the supremum over all sequences of the ratio $b_{off}/E(b_{on})$. To prove the lower bound we apply a variation of Yao's theorem to the competitive ratios under consideration (see for example [ABM93] for this variation). This allows us to replace randomness in the algorithm with randomness in the input. We will choose a distribution on the input sequence such that the expected competitive ratio of any deterministic algorithm is at least the desired lower bound when averaging over the possible sequence inputs. We should emphasize that the correct interpretation of Yao's theorem requires us to compute the value $1/E(b_{on}/b_{off})$ over the probability distribution of the input sequence for any deterministic on-line algorithm.

We first find integer numbers y and z such that $n \geq 2^y z$ and $m \geq 2^z 2^y$. We construct the following probability distribution over the sequences. Choose an integer $1 \leq i^* \leq z$ uniformly at random. Construct sequences that consists of i^* phases while each phase consists of y steps. In phase i step j where $0 \leq i < i^*$, $1 \leq j \leq y$ there are elements for 2^{y-j} sets, where for each such set there are 2^i requests. More specifically , there are 2^{y-1} sets in the first step of a phase and they are different from all previous sets. The sets requested at step $j > 1$ of a phase are a random set of size 2^{y-j} out of the 2^{y-j+1} sets of the previous step. In other words it is a random set of half the size of the previous set.

Note that the number of new sets in each phase is precisely 2^{y-1} and there are at most z phases. Thus the number of sets is at most $2^{y-1}z \leq n$ sets. Moreover, the number of elements requested in phase i is less than $2^y 2^i$. Hence the total number of elements is bounded by $2^y 2^z \leq m$ This justifies the choices of y and z as a function of m and n.

We first observe that if the value i^* was chosen then the revenue of the off-line on any such sequence is $2^{i^*-1}y$. Consider any deterministic algorithm. Clearly, it has to choose which set to accept based only on the history. For a fixed history, assume that it chooses to accept some set at phase i' and at step j'. The probability that a sequence that coincides with the prefix of this this history will reach phase i' is $\frac{z-i'}{z}$. Then conditional probability that $i^* = i' + s$ for $0 \leq s < z - i'$ is $\frac{1}{z-i'}$. The probability that this set will stay a live for at least r additional steps is at most 2^{-r}. Thus the expected ratio between the profit of the

on-line algorithm to the profit of the off-line algorithm is at most

$$\left(\frac{z-i'}{z}\right)\cdot\left(\frac{\frac{2^{i'}}{(z-i')}+\frac{2^{i'+1}}{(z-i')}+\cdots\frac{2^{i^*-1}}{(z-i')}}{2^{i^*-1}}\right)\cdot\left(\frac{2^{-0}+2^{-1}+2^{-2}+\ldots}{y}\right)\leq\frac{4}{yz}$$

This implies the $yz/4$ lower bound. Note that in term of n and m the lower bound is $\log n \log m$ for wide range of values.

References

[ABFR94] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosén. Competitive Non-Preemptive Call-Control. In *Proc. of the 5th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 312-320, January 1994.

[ABF93] B. Awerbuch, Y. Bartal, and A. Fiat. Competitive Distributed File Allocation. In *Proc. of the 25th Ann. ACM Symp. on Theory of Computing*, pages 164-173, May 1993.

[ABM93] Y. Azar, A. Broder, and M. Mannase. On-line choice of on-line algorithms. In *Proc. 4th ACM-SIAM Symp. on Discrete Algorithms*, pages 432–440, 1993.

[AGH94] A. Aggarwal, Juan Garay, and Amir Herzberg. Adaptive video on demand. In *Proc. Thirteenth ACM PODC Symp.*, page 402, 1994. also appeared as an IBM Research Report, RC19770, Oct, 1994.

[CEL93] J. Cooperstock, R. El-Yaniv, T. Leighton. The Statistical Adversary Allows Online Foreign Exchange with no Risk. Proceedings of SODA '95.

[CV92] B. Chandra and S. Vishwanathan. Constructing Reliable Communication Networks of Small Weight On-line. Journal of Algorithms, 1992.

[EFKT92] R. El-Yaniv, A. Fiat, R. Karp, and G. Turpin. Competitive Analsys of Financial Games. In *Proc. of the 33th Ann. IEEE Symp. on Foundations of Computer Science*, pages 327-333, October 1992.

[EK93] R. El-Yaniv and R. Karp. The Mortage Problem. In *Proc. of the 2nd Ann. Israeli Symp. on Theoretical Computer Science*, May 1993.

[FKT89] U. Faigle, W. Kern and György Turán. On the Performance of On-Line Algorithms for Partition Problems. *Acta Cybernetica* 9, pages 107-119, 1989.

[HS92] M.M. Halldórsson and M. Szegedy. Lower Bounds for On-Line Graph Coloring. In *Proc. of the 3rd Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 211-216, January 1992.

[Irani90] S. Irani. Coloring Inductive Graphs On-Line. In *Proc. of the 31st Ann. IEEE Symp. on Foundations of Computer Science*, pages 470-479, October 1990.

[IW91] M. Imase and B.M. Waxman. Dynamic Steiner Tree Problem. In *SIAM Journal on Discrete Mathematics*, 4(3):369-384, August 1991.

[KMRS88] A.R. Karlin, M.S. Manasse, L. Rudolph, and D.D. Sleator. Competitive Snoopy Caching. In *Algorithmica*, 3(1):79-119, 1988.

[KVV90] R.M. Karp, U.V. Vazirani, and V.V. Vazirani. An Optimal Algorithm for On-Line Bipartite Matching. In *Proc. of the 22rd Ann. ACM Symp. on Theory of Computing*, pages 352-358, May 1990.

[LT94] Richard J. Lipton and Andrew Tomkins. Online interval scheduling. In *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms*, pages 302–311, Arlington, VA, January 1994.

[ST85a] D.D. Sleator and R.E. Tarjan. Amortized Efficiency of List Update and Paging Rules. In *Communications of the ACM*, 28(2) pages 202-208, 1985.

[ST85b] D.D. Sleator and R.E. Tarjan. Self-Adjusting Binary Search Trees. *Journal of the ACM*, 32:652–686, 1985.

[Vish90] S. Vishwanathan. Randomized Online Graph Coloring. In *Proc. of the 31st Ann. IEEE Symp. on Foundations of Computer Science*, October 1990.

[WY93] J. Westbrook. and D.K. Yan. Greedy On-Line Steiner Tree and Generalized Steiner Problems. In *Proc. of the 3rd Workshop in Algorithms and Data Structures*, Also *Lecture Notes in Computer Science*, vol. 709, pages 622-633, Montréal, Canada, 1993, Springer-Verlag.