On-line Complexity of Monotone Set Systems

(extended abstract)

Haim Kaplan *and* Mario Szegedy
AT&T Labs Research
{hkl,ms}@research.att.com

Abstract

On-line analysis models a player A (randomized or deterministic) who makes immediate responses to incoming elements of an input sequence $s = a_1 \ldots a_r$. In this paper $a_1, \ldots, a_r$ are interpreted as elements offered without repetition to the player from a fixed universe and the player's response to each a_i is a single bit interpreted as pick/not-to-pick. Before seeing the stream s, the player is given a monotone system $\mathcal{M}$ of sets indicating all sets of elements that she is allowed to hold at any time during the game and the player's objective is to pick as many elements as possible under these constraints. We assign the performance measure $e(\mathcal{M}) = \max_A \min_s E(|A(s)|)/OPT(s)$ to every monotone set system $\mathcal{M}$, where $E()$ is the expectation over the player's random choices and $OPT(s) = \max_{M \in \mathcal{M}} |M \cap s|$. Note that for every $\mathcal{M}$, $e(\mathcal{M}) \in [0, 1]$, and it is easy to show that a system has performance one iff it is a matroid. This model generalizes the apparently unrelated models in [2] and [3].

The video-on-demand problem studied in [1, 2] corresponds to the case when $\mathcal{M}$ is the monotone closure of l disjoint sets of size k. Let $e(k, l)$ denote the performance of this set system. Awerbuch et al. [1] show that $e(k, l) = \Omega(1/(\log k \log l))$. We improve this bound for some values of k and l by showing that $e(k, 2) = k/(2k - 1)$, $e(k, 3) = k/(3k - \Theta(\sqrt{k}))$, and $e(k, l) \geq k^{l-1}/(k^l - (k-1)^l)$. In particular, for any fixed l our bounds tend to $1/l$ as k grows whether the bound of Awerbuch et al. tends to zero. We also show, that if $\mathcal{M} = \mathcal{M}_1 \cup \ldots \cup \mathcal{M}_l$, and k is an upper bound on the size of the largest set in $\mathcal{M}$, then the performance of $\mathcal{M}$ is at least as good as the performance of the worst of the components multiplied by a factor of $e(k, l)$.

Our model also generalizes one of Bartal et al. [3] who considered set systems that arise from hereditary graph properties. Bartal et al. use specific graph product to prove that for every n there exists a graph G with $O(n)$ vertices such that the performance of the system of its independent sets is no greater than $n^{-0.207}$. Unfortunately their argument contains a flaw. We generalize the graph product to set systems, define the *conditional performance* of a set system and prove that the conditional performance of the product of set systems is bounded by the product of their conditional performances. As a corollary of this theorem we obtain a correct proof to Bartal et al.'s upper bound and even slightly improve it to $n^{-0.226}$ using a better base graph.

1 Introduction

An online algorithm receives a series of requests and must respond to each request immediately without any knowledge of future requests. In contrast, an off-line algorithm knows the complete request sequence before it starts serving the requests.

The standard measure for the performance of an online algorithm is the maximum of the ratio of the online and the optimal off-line solutions, where the maximum is taken over all input sequences. This ratio, called by Sleator and Tarjan [13] the *competitive ratio*, has been used to analyze algorithms for various data structures, paging, caching, and graph problems (See e.g. [5, 12, 8, 6]), and gave rise to elegant generalizations, such as metrical task systems introduced by Borodin, Linial, and Saks [4], and the K-server problems introduced by Manasse, McGeoch and Sleator [11] (see also [9]).

Most on-line models assume that there is small common knowledge shared between the online algorithm and the adversary. In particular, online models for graph problems assume that the graph is not known to the online algorithm. An exception to this is the model of Halldórsson in [6], in which the algorithm receives a graph in the start of the game that contains the future on-line input sequence (which is a polynomial fraction of this graph), but the adversary does not identify the upcoming vertices in the large graph (he reveals only their connections to previously shown vertices).

Bartal, Fiat and Leonardi [3] introduced recently a model for graph optimization problems, in which the on-line sequence consists of vertices of a fixed graph G known to the algorithm. The algorithm can choose either to pick or not to pick a vertex with the following provisions. The description of the game, besides G, includes a graph property Π, and the set of elements picked by the algorithm at any time during the game must have property Π. The algorithm does not know when the input sequence ends, so it should optimize its strategy for all possible lengths. In [3] Bartal et al. claim without a proof that for every nontrivial hereditary graph property Π and integer n there exists a graph with $O(n)$ vertices such that the performance of any online algorithm playing on G is at most $n^{-.337}$. More recently they have withdrawn their claim for this bound [10]. They do provide a proof in [3] for a weaker bound of $n^{-.207}$ but unfortunately their proof contains an error. We describe this error in detail in an appendix of an expanded version of this paper [7].

In this paper we define a generalization of this model, which also turns out to be the generalization of another interesting, and seemingly unrelated problem, known as the Video-on-Demand (VOD) scheduling problem [2, 1]. In this model customers issue requests for movies to a video server. Upon arrival, a movie request is accepted and served (possibly with some delay) or rejected. The goal of the video server is to maximize the number of accepted requests subject to a capacity constraint, which is the maximum number of movies that can be shown in parallel.

Our Model: A set system $\mathcal{M}$ on a basic set $B = B(\mathcal{M})$ is said to be *monotone* if for every $M \in \mathcal{M}$ and $M' \subseteq M$ we also have $M' \in \mathcal{M}$. Let $\mathcal{M}$ be a set system on a basic set $B(\mathcal{M})$. We shall assume, that $B(\mathcal{M}) = \bigcup_{M \in \mathcal{M}} M$. If $\mathcal{M}$ is not monotone, we can consider its *downward closure*, $\mathcal{M}^c$, which is the smallest monotone system containing $\mathcal{M}$. For our further discussion we assume that $\mathcal{M}$ is monotone, and we call its elements *independent sets*. Let $s = a_1 a_2 \dots a_r$, where $a_1, \dots, a_r \in B(\mathcal{M})$, $a_i \neq a_j$ for $1 \leq i \neq j \leq r$. We define $OPT(s, \mathcal{M})$, or shortly $OPT(s)$ to be the size of the maximum cardinality subset of s that belongs to $\mathcal{M}$. Notice, that for every non-empty sequence s, $OPT(s) \geq 1$. We shall denote the set of all possible input sequences, including the empty sequence ϵ, by $S(B)$.

Assume that A is a randomized online algorithm that picks or rejects elements of a sequence s in real time. If the set of elements picked by A for every input sequence and for every random choice of A belongs to $\mathcal{M}$, then we say that A *admits* $\mathcal{M}$. In the following definitions we consider only algorithms that admit $\mathcal{M}$. If $A(s)$ is the sequence of elements that A picks for an input sequence s, then $y(A, s) = E(|A(s)|)$ is the *yield* of A on s, $e(A, s) = y(A, s)/OPT(s)$ is the *performance of A on s* , and $e(A) = \min_{s \neq \epsilon} y(A, s)/OPT(s)$ is the *performance* of A. The *performance* of $\mathcal{M}$ is $\max_A e(A)$, the performance of the best performing algorithm that admits $\mathcal{M}$. We remark that the on-line performance of an algorithm is the reciprocal of what is called *competitive ratio* in the literature. We choose to state our results in terms of the on-line performance, for it is always between 0 and 1, and its increase well reflects the increasing success of the algorithm against the adversary. The fundamental issue we address here is how various properties of $\mathcal{M}$ affect the value of $e(\mathcal{M})$?

If $\mathcal{M}$ is the collection of subsets of the vertex set of a graph G satisfying some hereditary property, then we get the model of Bartal et al. On the other hand, if $\mathcal{M}$ is the downward closure of a system with disjoint maximal sets, then we get the Video-on-Demand scheduling problem posed in [2].

Our Results: We define composition of set systems and a new performance measure of set systems, called the *conditional performance*. We show that the product of the conditional performances of set systems upper bounds the conditional performance of their composition. The $n^{-.207}$ upper bound of Bartal et al. (which is incorrectly proven in their paper) follows as corollary from our composition theorem. Moreover with a different base graph we improve this bound to $n^{-.226}$.

Let $\mathcal{M}_{k,l}$ denote the downward closure of l disjoint sets each of cardinality k, and let $e(k, l)$ denote its performance. Awerbuch et al. [1] give an algorithm for a generalization of the VOD problem which implies that $e(\mathcal{M}_{k,l}) = \Omega(1/(\log k \log l))$. This bound is tight when k and l are not super-exponentially related [1]. We prove in Section 3.3 that $e(k, l) \geq k^{l-1}/(k^l - (k-1)^l)$ which is asymptotically better than $\Omega(1/(\log k \log l))$ for another range of parameters.

The simple structure of $\mathcal{M}_{k,l}$ makes us hope that computing $e(k, l) = e(\mathcal{M}_{k,l})$ exactly as a function of k and l is possible. Though this ultimate goal is still beyond our reach at this point, we have been able to make significant progress towards a complete understanding of these systems. In Section 3 we show that $e(k, 2) = k/(2k - 1)$. The $l = 3$ is more difficult to analyze. We prove that $k/(3k - 0.5 - \min_{z=1,\dots,k}(z/2 + k/z)) \leq e(k, 3) \leq k/(3k - \min_{z=1,\dots,k}(z + k/z))$, and describe a computer experiment that leads us to conjecture that the above lower bound is tight for $k > 4$. For practically occurring on-line problems finding the best performance can be done with a computer if the instance is very small. Otherwise, the search space has to be restricted without loosing the best algorithms. For $\mathcal{M}_{k,l}$ we restrict the search to a subset of online algorithms which we call *oblivious*. We conjecture that this class contains the best algorithm for any k and l.

One motivation of computing $e(k, l)$ precisely is a decomposition theorem, called the Union Theorem, which we prove in Section 4.3. The Union theorem upper bounds the performance of the union of l systems with independent sets of size at most k in terms of $e(k,l)$ and the performance of the components. In section 4.1 we also prove several other decomposition theorems that bound the performance of a set system in terms of the performances of smaller subsystems. In Section 4.2 we show that the performance of the union of set systems is at least as good as the performance of the disjoint union of isomorphic systems.

Bartal Fiat and Leonardi prove that $1/(2\sqrt{n}) \leq e(\mathcal{M})$ for any $\mathcal{M}$ that asrises from an hereditary graph property, where $n = |B(\mathcal{M})|$. It is straightforward to generalize their lower bound to any monotone set system. Moreover, in Section 4.1, as an application of one of our decomposition theorems, we improve this lower bound to $1/\sqrt{2n}$.

Although we are far from the goal of predicting $e(\mathcal{M})$ from the structure of the set system $\mathcal{M}$ in general, we have made a slight progress in this direction as well. In Section 6 we show that the set systems with $e(\mathcal{M}) = 1$ are exactly the matroids. We define extensions of matroids, and show, that they are characterized by certain types of greedy algorithms. We also give a combinatorial characterization of those elements in a set system (not necessarily a matroid) that any algorithm can greedily pick without hurting its performance.

Some proofs are omitted from this extended abstract. An expanded version of this paper is avaliable [7].

2 Preliminaries

Formal definition of an on-line algorithm: For a set system $\mathcal{M}$ we define $\Sigma = \{(s, M) \mid s \in S(B(\mathcal{M})), M \in \mathcal{M}, M \subseteq s\}$, and for a sequence s let $last(s)$ be the last element of s. An on-line algorithm is characterized by a function $p : \Sigma \to [0,1]$ satisfying the following conditions. (1) $p(\epsilon, \emptyset) = 1$; (2) For every $(s, M) \in \Sigma$, $p(s, M) \geq 0$; (3) For every $(s, M) \in \Sigma$ such that $last(s) \notin M$, if $M \cup \{last(s)\} \notin \mathcal{M}$, $p(s, M) = p(s \backslash \{last(s)\}, M)$, and if $M \cup \{last(s)\} \in \mathcal{M}$, $p(s, M) + p(s, M \cup \{last(s)\}) = p(s \setminus \{last(s)\}, M)$. The value $p(s, M)$ is the **absolute probability** that A picks a set $M \in \mathcal{M}$ for an input sequence s.

Since conditions (1), (2), and (3) above are linear, the online algorithms occupy a convex polytope **ON** in the Σ dimensional space. It is intuitively straightforward that the vertices of **ON** correspond to deterministic algorithms. We omit the formal proof of this fact from this abstract. We can also express the on-line performance of an on-line algorithm with respect to an input sequence s as

$$(2.1) \qquad e(A, s) = \sum_{M \in \mathcal{M}, M \subseteq s} \frac{|M|}{OPT(s)} p(s, M).$$

Yao's Lemma: The often quoted Lemma of Yao [15] relates the performance of randomized algorithms to the performance of deterministic algorithms on random inputs. In order to state Yao's lemma for our model we extend some functions defined in Section 1 on input sequences to probability distributions on input sequences. Let D be an arbitrary distribution on input sequences from $S(B(\mathcal{M}))$ Then $OPT(\mathcal{M}, D) = \sum_{s \in S(B(\mathcal{M}))} D(s) OPT(s)$, and for an on-line algorithm A admitting $\mathcal{M}$, $y(A, D) = \sum_{s \in S(B(\mathcal{M}))} D(s) y(A, s)$.

THEOREM 2.1. (YAO'S LEMMA) $e(\mathcal{M}) = \min_D \max_A (y(A, D))/OPT(D))$, *where the minimum is over all probability distributions D on $S(B)$ and the maximum is over all deterministic online algorithms A admitting $\mathcal{M}$.*

The pick function: Let A be an online algorithm admitting a set system $\mathcal{M}$. We define a function $pick_A()$ from $S(B(\mathcal{M}))$ to $[0,1]$ assigning to each sequence the probability that $A(s) \neq \emptyset$ (If A is deterministic $pick()$ takes only the values 0 and 1). We shall omit the subscript A when it is clear from the context. We say that $pick()$ is *oblivious* if it assigns the same value for every two sequences which contain the same set of elements. The pick function plays an important role in Section 3 where it completely characterizes an online algorithm for a system with disjoint maximal sets. It also plays an important role in Section 5 where we define a the conditional performance of a set system that multiplies when we compose set systems.

3 Systems with Disjoint Maximal Sets

Let $\mathcal{M}_{k,l} = \{H_1, \ldots, H_l\}^c$ be a system such that $H_i \cap H_j = \emptyset$ for $1 \leq i \neq j \leq l$, and $|H_i| = k$ for $1 \leq i \leq l$. For an $a \in B(\mathcal{M}_{k,l})$ we denote by $H(a)$ the maximal independent set that contains a. We denote $e(\mathcal{M}_{k,l})$ by $e(k,l)$.

The value of $e(k,l)$ gives the performance of the best algorithm for the basic case of the online Video On Demand problem, and plays a crucial role in our union theorem of the next section. From the result of Awerbuch et al. we have that $e(k,l) = \Omega(1/(\log k \log l))$.

When we started doing research on this problem our goal was to find the best algorithm admitting $\mathcal{M}_{k,l}$ for every k and l. Although we haven't yet achieved that we have the following results.

1) We give an explicit formula for $e(k,2)$ and $e(2,l)$.

2) We give very tight upper and lower bounds on $e(k,3)$.

We conjecture that our lower bound is optimal based on computer experiments. Generalizing this result to $\mathcal{M}_{k,4}$ is already an intriguing open question.
3) We describe a new algorithm admitting $\mathcal{M}_{k,l}$ whose performance is $k^{l-1}/(k^l-(k-1)^l)$ which is asymptotically better than $\Omega(1/(\log k \log l))$ for some values of k and l.

The definition of the pick function in Section 2 is particularly interesting in the context of $\mathcal{M}_{k,l}$. For $\mathcal{M}_{k,l}$ the pick function completely characterizes the online algorithm: Once the algorithm picks an element, it commits itself to a particular maximal set. Conversely, any function $pick()$ from $S(B(\mathcal{M}_{k,l}))$ to the reals such that (1) $pick(\emptyset)=0$; (2) $\forall\ s\in S(B):\ \ 0\le pick(s)\le 1$; (3) $\forall\ s\in S(B),\ x\in B\backslash s:\ \ pick(s)\le pick(sx)$; defines a unique on-line algorithm A which admits $\mathcal{M}_{k,l}$. We can also specify the yield of an algorithm A on a sequence $s=a_1\ldots a_r$ as follows. Let s^j, $0\le j\le k$ be the prefix of s of length j and let $\Delta pick(s^j)=pick(s^j)-pick(s^{j-1})$ then $y(s)=\sum_{i=1}^r \Delta pick(s^j)|(s\setminus s^{j-1})\cap H(a_j)|$.

We say that an algorithm for $\mathcal{M}_{k,l}$ is *oblivious* if it corresponds to an oblivious pick function. If A is an algorithm for $\mathcal{M}_{k,l}$ then $\sum_{\sigma\in \mathrm{Aut}(\mathcal{M})}\sigma A\sigma^{-1}/|\mathrm{Aut}(\mathcal{M})|$ is also an algorithm admitting $\mathcal{M}_{k,l}$ whose performance is at least as good as the performance of A. Hence we may assume that a pick function defining an oblivious algorithm depends only on the multi-set of "multiplicities" $m_1=|s\cap H_1|,\ldots,m_l=|s\cap H_l|$.

The algorithm of Awerbuch et al. can be described by the following pick function.

$$pick(s)=1-\frac{1}{\lceil\log k\rceil+1}\sum_{i=1}^{\lceil\log k\rceil}\prod_{j=1}^{OPT(s)}\chi\left(1-\frac{2^{j/2^i}}{2^i l^2}\right)^{w_j}. \quad (3.2)$$

where $w_i=|\{1\le j\le l \mid m_j>=i\}|$ and $\chi(x)$ equals x if $x\ge 0$ and 0 otherwise. Awerbuch et al. show that the performance of this algorithm is $\Omega(1/\log k\log l)$. Note that this pick function is oblivious. Moreover all algorithms in this section are oblivious. This observation leads us to the following Conjecture.

Conjecture 3.1. *There is an oblivious algorithm $A_{k,l}$ such that $e(A_{k,l})=e(k,l)$.*

3.1 Two disjoint sets and sets of pairs.

The following theorem shows that $e(k,2)=k/(2k-1)$.

Theorem 3.1. *Let $\mathcal{M}$ be a set system consisting of two disjoint maximal sets H_1 and H_2, such that $|H_1|\le|H_2|=k$. Then $e(\mathcal{M})=k/(2k-1)$.*

We can similarly characterize the performance of systems with l disjoint maximal sets each of size two.

Theorem 3.2. $e(2,l)=l/(2l-1)$.

3.2 The case of Three disjoint sets.

In this section we describe an **oblivious** algorithm admitting $\mathcal{M}_{k,3}$ whose competitive ratio is $f(k)$ where $f(k)=k/(3k-\frac{1}{2}-\min_{z=1,\ldots,k}(z/2+k/z))$, for $k\ne 2,4$, $f(2)=3/5$, and $f(4)=13/28$. We remark that $\min_{z=1,\ldots,k}(z/2+k/z)$ is obtained either for $z=\lfloor\sqrt{2k}\rfloor$ or for $z=\lfloor\sqrt{2k}+1\rfloor$, so $f(k)\approx\frac{k}{3k-1/2-\sqrt{2k}}$. Our main theorem is the following

Theorem 3.3. $e(k,3)\ge f(k)$.

We also have the following upper bound.

Theorem 3.4. $e(k,3)\le k/(3k-min_{z=1,\ldots,k}(z+k/z))\approx\frac{k}{3k-2\sqrt{k}}$

To prove Theorem 3.3 we shall construct an algorithm admitting $\mathcal{M}_{k,3}$ and show that its performance is $f(k)$. Intuitively, any algorithm for large k cannot do much better than the algorithm that decides uniformly at random on a set $H\in\{H_1,H_2,H_3\}$ and then picks every element from H. Therefore, we expect the performance of $\mathcal{M}_{k,3}$ to tend to 1/3 as k grows. Indeed, our algorithm gets similar to the algorithm above as k tends to infinity.

It is feasible to find the best oblivious algorithm for $\mathcal{M}_{k,l}$ by computer when k and l are small. The idea is to generate a linear expression for the on-line performance of the algorithm with respect to every possible input sequence, where the variables in these expressions are the values of the function $pick()$. Then by solving a linear program we find the function $pick()$ that maximizes the value of the minimum expression. We have implemented this experiment and verified that our algorithm is the best oblivious algorithm for $\mathcal{M}_{k,3}$ for every $k\le 10$. This experiment together with Conjecture 3.1 leads us to conjecture that we have found the best algorithm for every k, i.e. we conjecture that $e(k,3)=f(k)$.

Proof. [of Theorem 3.3] Consider the algorithm defined by the following pick function admitting $\mathcal{M}_{k,3}$. We specify pick() in terms of three parameters $0\le e,x,y\le 1$. Let s be a sequence such that $|s\cap H_i|=a$, $|s\cap H_j|=b$, $|s\cap H_k|=c$, where $\{i,j,k\}=\{1,2,3\}$ and $c\le b\le a$. Then $pick(s)=\min(1,e+2x+(a+b+c-3)y)$ if $c>0$, $pick(s)=\min(1,e+x+(a+b-2)y)$ if $c=0$ and $b>0$, and $pick(s)=\min(1,e+(a-1)y)$ if $c=0$ and $b=0$. The parameters are set as follows, $e=f(k)$ for every k, for $k\ne 2,4$, (1) $y=e/k$ and (2) $x=1-e-(k-1)y$, whereas for $k=2$, $y=1/5$ and $x=1/5$, and for $k=4$, $y=3/28$ and $x=6/28$. We shall prove that this algorithm has performance $f(k)$. For $k=2,4$ it is possible to verify that the claim holds by calculating the performance for every input sequence. Therefore in the rest of the proof we assume that $k \neq 2, 4$.

We shall show that it suffices to guarantee performance of at least $f(k)$ for the sequences $s_l = as'_l$, $0 \leq l \leq k$, where $a \in H_1$ and $s'_l \subseteq H_2$, $|s'_l| = l$. Since $e = f(k)$ we want to guarantee that $e(s_l) \geq e$ for $0 \leq l \leq k$. Clearly $e(s_0 = a) = e$. For $l > 0$, by writing $e(s_l)$ in terms of x, y, and e we obtain that we want $(e + lx + l(l-1)y/2)/l \geq e$ to hold for every $1 \leq l \leq k$. Using Formula (1) and (2) we can rewrite this condition as $y \leq 1/(3k - 1/2 - (l/2 + k/l))$. The latter inequality holds by our settings of e and y.

In the rest of the proof we show that indeed it suffices to guarantee the claimed performance for the sequences s_l, We prove this in three stages. Stage 1 shows that for every sequence s there exists a sequence $s_1 = a_1 \ldots a_r$ such that $s_1 \cap H_1 = \{a_1\}$, $a_r \in H_2$, $OPT(s_1) = |s_1 \cap H_2|$, and $e(s_1) \leq e(s)$. Stage two shows that for every such s_1 there exists s_2 such that $s_2 = a_1 s' s''$ where $a_1 \in H_1$, $s' \subseteq H_3$, $s'' = H_2$, and $e(s_2) \geq e$ implies $e(s_1) \geq e$. Stage three shows that for every such s_2 there exists $s_3 = a_1 s'$ where $a_1 \in H_1$, $s' \subseteq H_2$, and $e(s_3) \geq e$ implies $e(s_2) \geq e$. The proofs of each of these three stages follow.

1) We may assume that the first element in the input sequence is from H_1. If $OPT(s) = |s \cap H_1|$ then $e(s) \geq e = f(k)$ as required so using the symmetry again we may also assume that $OPT(s) = |s \cap H_2|$

Assume $s \cap H_1 \neq \{a_1\}$. Let $s_1 = a_1 \ldots a_r$ be the sequence obtained from s by deleting all the elements from H_1 except a_1. We claim that $e(s_1) \leq e(s)$. Since $OPT(s_1) = OPT(s)$ in order to prove the claim it suffices to show that $y(s_1) \leq y(s)$. Let s_1^i, $0 \leq i \leq r$ be the prefix of s_1 of length i, and let s^i be the prefix of s that ends with the same element as s_1^i (i.e. s^i need not be of length i). We have that $y(s_1) = e + \sum_{i=2}^{r} \Delta pick(s_1^i) \left|(s \setminus s'_{i-1}) \cap H(a'_i)\right|$. Since $\sum_{i=2}^{r} \Delta pick(s_1^i) \leq \sum_{i=2}^{r} \Delta pick(s_i) + |(s - \{a_1\}) \cap H(a_1)| y$ and $ky = e$ we obtain that $y(s_1) \leq |s \cap H_1| e + \sum_{i=2}^{r} \Delta pick(s_i) \left|(s \setminus s_1^{i-1}) \cap H(a_i)\right| \leq y(s)$.

We may also assume that the last element of s_1 belongs to H_2 since otherwise by deleting this element we obtain a sequence with smaller performance. This completes the proof of stage one.

2) Let $s_1 = a_1 \ldots a_r$ such that $s_1 \cap H_1 = \{a_1\}$, $a_r \in H_2$, $s_1 \cap H_3 \neq \emptyset$, and $OPT(s_1) = |s_1 \cap H_2|$. If $OPT(s_1) < k$ let $s'_1 = s_1 b$ where $b \in H_2 \setminus s_1$. Then $y(s'_1) \leq y(s_1) + (1 - e - x) = y(s_1) + (k-1)e/k$, and $OPT(s'_1) = OPT(s_1) + 1$. Thus if $eOPT(s'_1) \leq y(s'_1)$ then $eOPT(s_1) \leq y(s_1)$. Therefore we may assume that $s_1 \cap H_2 = H_2$.

If the elements from H_2 form one consecutive block then s_1 already has the form required by stage two so we can set $s_2 = s_1$. Otherwise, if $|s_1 \cap H_3| > 1$, let a_l be the last element from H_3 in s_1, and let s'_1 be the sequence obtained from s by deleting a_l. We have that $y(s_1) - y(s'_1) = \sum_{a_j \in H_3} \Delta pick(s_1^j) + \sum_{j=l+1}^{r} \Delta pick(s_1^j) - y(r - l + 1) = pick(s_1) - \sum_{a_j \in H_2, j < l} \Delta pick(s_1^j) - y(r - l + 1) = pick(s_1) - e - x - (OPT(s_1) - 1)y \geq 1 - e - x - (k-1)y = 0$. By iterating this argument we end up either with a sequence s'_1 in which the elements from H_2 form one consecutive block at the end or with a sequence s'_1 such that $|s'_1 \cap H_3| = 1$. In former case s'_1 has the form required by stage two so we set $s_2 = s'_1$. In the latter case the sequence s_2 is obtained from s'_1 by shifting the single element from H_3 to be the second in the sequence. A straightforward calculation shows that $e(s_2) \leq e(s'_1)$.

3) Let $s_2^j = a_1 s^j s''$ where $a_1 \in H_1$, $s^j \subseteq H_3$, $|s_j| = j$, $j > 0$, and $s'' = H_2$. Let $l = \max\{i | e + x + iy > x\}$. A straightforward calculation (that takes (2) into account) shows that $y(s_j) = y(s_{j+1})$ for $j = 1 \ldots l$. If we assume that $y(a_1 s^j) \geq je$ for $1 \leq j \leq k$ then we have that for $j = (l+1) \ldots k$, $y(s_2^j) = y(as^j) + (1 - e - x - (j-1)y)k \geq ke$.

3.3 A general lower bound.

Below we give a lower bound on $e(k, l)$, that is better than the one in given by Awerbuch et al. [1] for some values of k and l.

THEOREM 3.5. $e(k,l) \geq k^{l-1}/(k^l - (k-1)^l)$.

Proof. We design algorithms $A_{k,l}$ with performance at least $f(k,l) = \frac{k^{l-1}}{k^l - (k-1)^l}$ recursively. Let $A_{k,1}$ be the greedy algorithm for every $k \geq 1$. For $l \geq 2$ let $A_{k,l}$ be the algorithm that picks the very first element x of any input sequence s with probability $f(k,l)$. If $A_{k,l}$ picks the first element of s, then it also picks all subsequent elements from $H(x)$, otherwise it disregards these occurrences, and runs $A_{k,l-1}$ on $s \setminus H(x)$. For the analysis we use induction on l. We distinguish in between two cases:

$OPT(s) = |H(x)|$: In this case $A_{k,l}$ obviously achieves the required performance.

$OPT(s) \neq |H(x)|$: In this case $OPT(s) = OPT(s \setminus H(x))$ so $y(A_{k,l}, s) \geq f(k,l) + (1 - f(k,l)) y(A_{k,l-1}, s \setminus H(x))) \geq f(k,l) + (1 - f(k,l)) OPT(s) f(k, l-1) \geq \left(\frac{f(k,l)}{k} + (1 - f(k,l)) f(k, l-1)\right) OPT(s)$. Since $\frac{f(k,l)}{k} + (1 - f(k,l)) f(k, l-1) = f(k,l)$ we obtain $y(A_{k,l}, s) \geq f(k,l) OPT(s)$, from which the theorem follows.

4 Monotonicity and Decomposition Theorems

In this section we define several different ways to decompose a set system into simpler components that are usually set systems themselves. For each kind of decomposition we estimate the performance of the system in terms of the performances of the simpler components.

4.1 On-line Decomposition of the Sequence Space. In this section we define on-line decomposition of $S(B(\mathcal{M}))$ into sets of sequences $\mathcal{S}_1, \ldots, \mathcal{S}_l$. Theorem 4.1 relates the performance of $\mathcal{M}$ to the performance of $\mathcal{M}$ with respect to that of each of the components $\mathcal{S}_1, \ldots, \mathcal{S}_l$.

We shall denote the maximum length prefix of a sequence s that is in a set of sequences $\mathcal{S}$ by $pr(s, \mathcal{S})$. Let $\mathcal{S}_1, \ldots, \mathcal{S}_l$ be sets of sequences and let $A_1, \ldots, A_l$ be deterministic online algorithms. We say that $A_1, \ldots, A_l$ *decompose* $S(B)$ into $\mathcal{S}_1, \ldots, \mathcal{S}_l$ if $s = \bigcup_{i=1}^{l} pr(A_i(s), \mathcal{S}_i)$, for every $s \in S$.

For $\mathcal{S} \subseteq S(B(\mathcal{M}))$ we define the *online complexity of* $\mathcal{M}$ *with respect to* $\mathcal{S}$, $e(\mathcal{M}, \mathcal{S})$, as $\max_A \min_{s \in \mathcal{S}} e(A, s)$, where A runs through all randomized algorithms that admit $\mathcal{M}$. The following theorem shows that if we have a decomposition $\mathcal{S}_1, \ldots, \mathcal{S}_l$ of $S(B(\mathcal{M}))$ we can use $e(\mathcal{M}, \mathcal{S}_i)$, $1 \leq i \leq l$, to give a lower bound on $e(\mathcal{M})$.

THEOREM 4.1. *Let $A_1, \ldots, A_l$ be deterministic online algorithms that decompose $S(B(\mathcal{M}))$ into $S_1, \ldots, S_l$. Then, $e(\mathcal{M}) \geq 1/(1/e(\mathcal{M}, \mathcal{S}_1) + 1/e(\mathcal{M}, \mathcal{S}_2) + \ldots + 1/e(\mathcal{M}, \mathcal{S}_l))$.*

Proof. Let B_i, $1 \leq i \leq l$, be an online algorithm admitting $\mathcal{M}$ whose efficiency on $\mathcal{S}_i$ is $e(\mathcal{M}, S_i)$. It is straightforward to verify that the online algorithm A which chooses an index i, $1 \leq i \leq l$, with probability $e^{-1}(\mathcal{M}, \mathcal{S}_i)/\sum e^{-1}(\mathcal{M}, \mathcal{S}_i)$, and which then runs B_i on the elements picked by A_i admits $\mathcal{M}$, and has on-line performance as requested.

By *restricting* a set system $\mathcal{M}$ to a subset $B' \subseteq B$ one obtains another set system, $\mathcal{M}|B' = \{M \in \mathcal{M} \mid M \subseteq B'\}$. The following decomposition theorem is a corollary of Theorem 4.1.

COROLLARY 4.1. *Let $B_1 \cup B_2 \cup \ldots \cup B_l = B(\mathcal{M})$. Then $e(\mathcal{M}) \geq 1/[1/e(\mathcal{M}|B_1) + \ldots + 1/e(\mathcal{M}|B_l)]$.*

Proof. Let $\mathcal{S}_i = S(B_i)$ and let A_i be the deterministic online algorithm that picks x if and only if $x \in B_i$. Clearly $A_1, \ldots, A_l$ decompose $S(B)$ into $\mathcal{S}_1, \ldots, \mathcal{S}_l$. We finish the proof by applying Theorem 4.1 with this decomposition of $S(B)$.

COROLLARY 4.2. *Let $S(B(\mathcal{M})) \subseteq \mathcal{S}_1 \cup \ldots \cup \mathcal{S}_l$. Then $e(\mathcal{M}) \geq 1/[1/e(\mathcal{M}, \mathcal{S}_1) + \ldots + 1/e(\mathcal{M}, \mathcal{S}_l)]$.*

Proof. For every i let A_i be the algorithm that always picks every element. Then $A_1, \ldots, A_l$ decomposes $S(B)$ into $\mathcal{S}_1, \ldots, \mathcal{S}_l$, and the theorem follows.

The following theorem is similar to Corollary 19 in [3], but with a better constant factor.

THEOREM 4.2. *For every $\mathcal{M}$, $e(\mathcal{M}) \geq 1/\sqrt{2n}$, where $n = |B(\mathcal{M})|$.*

Proof. The proof is by induction on n. Every set system on two elements is a matroid and hence has performance one. Systems with three elements have performance at least $1/2$ since either the system is the trivial matroid containing all subsets or the optimum is at most two. Similarly, systems with four elements have performance at least $1/3$. Thus the statement is true for $n \leq 4$. Assume $n > 4$. If $OPT(\mathcal{M}) \leq \sqrt{2n}$, then the greedy algorithm gives $e(\mathcal{M}) \geq 1/\sqrt{2n}$. Otherwise, let $M \in \mathcal{M}$ with $|M| \geq \sqrt{2n}$. Notice, that $e(\mathcal{M}|M) = 1$, and by induction $e(\mathcal{M}|B \setminus \mathcal{M}) \geq 1/\sqrt{2(n - \sqrt{2n})}$. The statement now follows by applying Corollary 4.1.

4.2 The Performance of Disjoint Union Versus Union. It is natural to look for parameters of the set system on which the performance depends monotonically. For example the performance is not monotone with respect to the number of sets: as we add sets, the performance can increase as well as decrease. In this section we prove a monotonicity theorem that requires the notion of *splitting*.

DEFINITION 4.1. (SPLITTING) *Let $x \in B(\mathcal{M})$ and let $\mathcal{M}_x = \{M \in \mathcal{M} \mid x \in M\}$. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be arbitrary monotone systems such that $\mathcal{M}_x = \mathcal{M}_1 \cup \mathcal{M}_2$. From $\mathcal{M}_1$ we create $\mathcal{M}_1'$ by replacing x with $x_1 \notin B(\mathcal{M})$. Similarly, from $\mathcal{M}_2$ we create $\mathcal{M}_2'$ by replacing x with $x_2 \notin B(\mathcal{M}) \cup \{x_1\}$. The new system $(\mathcal{M} \setminus \mathcal{M}_x) \cup \mathcal{M}_1' \cup \mathcal{M}_2'$ is said to be obtained from $\mathcal{M}$ by splitting x into x_1 and x_2.*

It is easy to see that $(\mathcal{M} \setminus \mathcal{M}_x) \cup \mathcal{M}_1' \cup \mathcal{M}_2'$ is monotone for every x and every $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $M_x = \mathcal{M}_1 \cup \mathcal{M}_2$.

THEOREM 4.3. *Let $\mathcal{M}'$ be a system obtained from $\mathcal{M}$ by splitting x into x_1 and x_2. Then, $e(\mathcal{M}') \leq e(\mathcal{M})$.*

Proof. Let A' be an algorithm that admits $\mathcal{M}'$ and has performance $e(A') = e(\mathcal{M}')$. We construct an algorithm A that admits $\mathcal{M}$, and has performance at least $e(\mathcal{M}')$. Algorithm A simulates A' as follows:

Let y be the current element. If $y \neq x$, A sends y to A' and if A' picks y then so does A. If $y = x$, then A

sends x_1 followed by x_2 to A'. If A' picks x_1 or x_2 then A picks x, otherwise it takes no action and goes on to the next input element.

By definition 4.1, A admits $\mathcal{M}$. Let s' be the sequence that A' encounters while A processes s. Clearly, $OPT(s', \mathcal{M}') = OPT(s, \mathcal{M})$ and $y(A', s') = y(A, s)$, so $e(M') \leq e(A)$ and the statement follows.

Corollary 4.3. *If $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2$, and $\mathcal{M}' = \mathcal{M}'_1 \dot{\cup} \mathcal{M}'_2$ such that $\mathcal{M}'_1$ is isomorphic to $\mathcal{M}_1$ and $\mathcal{M}_2$ is isomorphic to $\mathcal{M}_2$, then $e(\mathcal{M}) \geq e(\mathcal{M}')$ ($\dot{\cup}$ stands for disjoint union).*

Proof. $\mathcal{M}'$ can be obtained from $\mathcal{M}$ by a sequence of element splits.

Example: Consider the set system that consists of the collinear points in a projective plane of rank q. Then Corollary 4.3 gives us that its performance is at least $e(q+1, q^2+q+1) = \Omega(1/\log^2 q)$. (We can also show that this is tight up to a constant factor)

4.3 The Union Theorem. We denote by $OPT(\mathcal{M})$ the size of the largest set in $\mathcal{M}$. Our main decomposition theorem is the following.

Theorem 4.4. (Union Theorem) *Let $\mathcal{M} = \mathcal{M}_1 \cup \ldots \cup \mathcal{M}_l$ be a set system such that $OPT(\mathcal{M}_i) \leq k$ for $1 \leq i \leq l$. Assume moreover that $e(\mathcal{M}_i) \geq \alpha$ for $1 \leq i \leq l$. Then, $e(\mathcal{M}) \geq \alpha e(k, l)$, where $e(k, l)$ is the performance of the system $\mathcal{M}_{k,l} = \{H_1, \ldots, H_l\}^c$ introduced in Section 3.*

Proof. Let A_0 be an algorithm admitting $\mathcal{M}_{k,l}$ such that $e(A_0) = e(k, l)$, and let B_i be an algorithm admitting $\mathcal{M}_i$ such that $e(B_i) \geq \alpha$ for every $1 \leq i \leq l$. Let $s = a_1 \ldots a_r$ be an on-line sequence, and let s^j be the prefix of s of length j.

Let A be the following algorithm admitting $\mathcal{M}$. Given s we construct (on-line) an input sequence s_0 for A_0. Initially s_0 is the empty sequence. When a_j arrives then for every i such that $OPT(s^j, \mathcal{M}_i) = OPT(s^{j-1}, \mathcal{M}_i) + 1$ we attach a new element from H_i to s_0. When A_0 picks an element say $x \in H_p$, we run B_p on the rest of s and pick every element picked by B_p. For the purposes of the analysis assume that s_0 keeps being constructed even when we started running B_p. Let s'_0 be the suffix of s_0 that start with x, and let a_l be the element from s that caused x to be sent to A_0. Using the assumptions on A_0 and B_p we have that $A(s) \geq \alpha E(OPT(s \setminus s^l, \mathcal{M}_p)) \geq \alpha E(|H_p \cap s'_0|) \geq \alpha OPT(s_0, \mathcal{M}_{k,l}) e(k, l) = \alpha OPT(s, \mathcal{M}) e(k, l)$

In particular the bound of Awerbuch et al. on $e(k, l)$ and the bound in Theorem 3.5 imply:

Corollary 4.4. *Let $\mathcal{M} = \mathcal{M}_1 \cup \ldots \cup \mathcal{M}_l$ be a set system such that the optimum of $\mathcal{M}_i$ is at most k for $1 \leq i \leq k$. Assume moreover that $e(\mathcal{M}_i) \geq \alpha$ for $1 \leq i \leq k$. Then $e(\mathcal{M}) \geq \alpha \Omega(1/(\log k \log l))$, and $e(\mathcal{M}) \geq \alpha k^{l-1}/(k^l - (k-1)^l)$.*

Example: We consider the following *On-line Monochromatic Forest problem.* In this problem the set system $\mathcal{M}$ is defined using a graph G whose edges are colored with colors from a color set C. The independent sets of $\mathcal{M}$ are the monochromatic forests of G. By Corollary 4.4 and Theorem 6.1 we obtain an on-line algorithm admitting $\mathcal{M}$ whose performance is at least $\Omega(1/(log|V(G)| \log |C|))$.

5 Compound Systems and Conditional Performance

Bartal et al. [3] invented a recursive construction based upon a specific graph product in order to obtain lower bounds on the competitive ratio. Their lower bound increases at the rate of some fixed root of the size of the construction, but their argument contains an error. In this section we generalize this construction, and identify the parameter that gets carried on to the higher levels of this construction.

Let $\mathcal{M}_0$ be a set system on basic set B_0 where $B_0 = \{1, \ldots, l\}$, and let $\mathcal{M}_1, \ldots, \mathcal{M}_l$ be set systems on basic sets $B_1, \ldots, B_l$. We assume that $B_i \cap B_j = \emptyset$ for $1 \leq i \neq j \leq l$. The *$\mathcal{M}_0$-composition of systems* $\mathcal{M}_1, \ldots, \mathcal{M}_l$ is a set system on $B = B_1 \cup B_2 \cup \ldots \cup B_l$ whose independent sets are all sets $I_{i_1} \cup I_{i_2} \cup \ldots \cup I_{i_s}$, where $\{i_1, i_2, \ldots, i_s\} \in \mathcal{M}_0$, and $I_{i_j} \in \mathcal{M}_{i_j}$ for $1 \leq j \leq s$.

In the remaining part of this paper we shall give weights to the elements of the basic set of a set system $\mathcal{M}$. Let w be a weight function from $B(\mathcal{M})$ to the non-negative reals. We extend our basic definitions from Section 1 to the weighted case as follows. The *yield*, $y(A, s)$, of an algorithm A for an input sequence s is the expected value of the total weight of the elements that A picks from s. We also define $OPT(s)$ to be the maximum weight of an independent set in s. The notions of *on line performance* of an algorithm and of a set system change accordingly. When we do not explicitly mention the weight function it is identically one.

We linearly extend the *pick*() function defined in Section 2 from input sequences to distributions on input sequences as follows. Let D be an arbitrary distribution on input sequences from $S(B(\mathcal{M}))$. Let A be an on-line algorithm that admits $\mathcal{M}$. Then we define $pick_A(D) = \sum_{s \in S(B(\mathcal{M}))} D(s) pick_A(s)$.

For a deterministic algorithm A for which

$pick_A(D) \neq 0$ we define $y'(A,D) = y(A,D)/pick_A(D)$. The *conditional performance* of a set system $\mathcal{M}$, $e'(\mathcal{M})$, is defined as $e'(\mathcal{M}) = \min_D \max_A (y'(A,D)/OPT(D))$, where A runs through all deterministic algorithms for which $pick_A(D) \neq 0$.

Our next theorem upper bounds the performance of compound systems in terms of the conditional performance of the components. It is unknown under what set of conditions we can exchange the conditional performance with performance in this theorem.

THEOREM 5.1. *Let $\mathcal{M}_1, \ldots, \mathcal{M}_l$ be arbitrary set systems such that $e'(\mathcal{M}_1), \ldots, e(\mathcal{M}_l) \leq \alpha$, and $\mathcal{M}_0$ be a weighted system such that $B(\mathcal{M}_0) = \{1, \ldots l\}$ and $w(x) = OPT(\mathcal{M}_x, D_x)$ for every $x \in B(\mathcal{M}_0)$, where D_i is the distribution that certifies that $e'(\mathcal{M}_i) \leq \alpha$. Then $e(\mathcal{M}_0(\mathcal{M}_1, \ldots, \mathcal{M}_l)) \leq e(\mathcal{M}_0)\alpha$ and $e'(\mathcal{M}_0(\mathcal{M}_1, \ldots, \mathcal{M}_l)) \leq e'(\mathcal{M}_0)\alpha$.*

Proof. We prove the first inequality. The proof of the second is similar. Let $B_i = B(\mathcal{M}_i)$. We define a distribution D on $S(B_1 \cup B_2 \cup \ldots \cup B_l)$ as follows. To generate a random sequence from $S(B_1 \cup B_2 \cup \ldots \cup B_l)$ according to D we first draw a random sequence $s = a_1 \ldots a_g$ from $S(B_0)$ according to D_0, and then for each a_i ($1 \leq i \leq g$) we draw a sequence from $S(B_{a_i})$ randomly according to the distribution D_{a_i}. The random sequence we construct is going to be the concatenated sequence $s(a_1)s(a_2)\ldots s(a_l)$. We will use the Lemma of Yao with this distribution to prove the theorem. Therefore in the followings A is arbitrary deterministic online algorithm admitting $\mathcal{M} = \mathcal{M}_0(\mathcal{M}_1, \ldots, \mathcal{M}_l)$ and our goal is to upper bound $y(A,D)/OPT(\mathcal{M},D)$.

For a fixed $x \in B_0$ let $y(A,D,x)$ be the expected number of elements of B_x that A picks, and let $P(x)$ be the probability that A picks an element from B_x. We claim that (i) $y(A,D,x) \leq P(x)y'(\mathcal{M}_x, D_x)$.

We prove (i) by constructing a randomized algorithm A_x that admits $\mathcal{M}_x$, has average performance $y(A,D,x)$ for distribution D_x, and $pick_{A_x}(D_x) = P(x)$. We define A_x as follows.

Let $s = a_1 \ldots a_f \in S(B_x)$ be an on-line input sequence for A_x. Algorithm A_x draws a sequence s' from D and run A on s' until it encounters the first element from B_x in s' (if we never encounter such an element, the algorithm terminates without picking anything). At that point A_x starts sending the elements of s to A picking the same elements that A picks.

If we draw s randomly from D_x and then run A_x on s, then the distribution of the output sets is exactly the same as when we draw s from D, run A on it, and intersect the result with B_x. This implies, that $P(x) = pick_{A_x}(D_x)$, and $y(A,D,x) = y(A_x,D_x) = pick_{A_x}(D_x)y'(A_x,D_x) = P(x)y'(A_x,D_x)$ so (i) follows.

By using (i) we have that $y(A,D)/OPT(\mathcal{M},D) = \sum_{x \in B_0} y(A,D,x)/OPT(\mathcal{M},D) \leq \sum_{x \in B_0} P(x)y'(\mathcal{M}_x,D_x)/OPT(\mathcal{M},D) = \sum_{x \in B_0} P(x)w(x)\alpha/OPT(\mathcal{M},D)$. It is straightforward to prove that $OPT(D_0,\mathcal{M}_0) \leq OPT(D,\mathcal{M})$. Moreover, we describe below an algorithm A_0 admitting $\mathcal{M}_0$ such that $y(A_0,D_0) = \sum_{x \in B_0} P(x)w(x)$. Combining these facts we obtain that $e(\mathcal{M}) \leq e(\mathcal{M}_0)\alpha$ as required.

Algorithm A_0 constructs online a sequence s', on which it simulates A as follows. Let x be the current element given to A_0. If the sequence seen by A_0 so far is a prefix of some sequence in the support of D_0, algorithm A_0 draws a random sequence $s(x) = b_1 \ldots b_f$ from D_x, adds it to s', and simulates the corresponding steps of A on it. If A picks at least one element from $s(x)$, then A_0 picks x, otherwise it does not. If A_0 observes that the input sequence cannot be extended to a sequence in the support of D_0 then it stops picking elements. It is straightforward to verify that A_0 admits $\mathcal{M}_0$ and indeed $y(A_0,D_0) = \sum_{x \in B_0} P(x)w(x)$.

COROLLARY 5.1. *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two monotone set systems. Let $l = |B(\mathcal{M}_1)|$, and $\mathcal{M}_2^1 \ldots, \mathcal{M}_2^l$ be disjoint copies of $\mathcal{M}_2$. Then $e(\mathcal{M}_1(\mathcal{M}_2^1, \ldots, \mathcal{M}_2^l)) \leq e(\mathcal{M}_1)e'(\mathcal{M}_2)$ and $e'(\mathcal{M}_1(\mathcal{M}_2^1, \ldots, \mathcal{M}_2^l) \leq e'(\mathcal{M}_1)e'(\mathcal{M}_2)$.*

Proof. Scale up the weights of the elements of M_1 by a factor of $OPT(M_2)$. This clearly does not effect the performance of M_1. The statement now follows from Theorem 5.1.

Let $ind(G)$ be the set system of all independent sets of a fixed graph G. Let G_0, G_1, ..., G_l be graphs on disjoint vertex sets such that $V(G_0) = \{1, 2, \ldots, l\}$, and let $G_0(G_1, \ldots, G_l)$ be the graph with $V(G) = V(G_1) \cup V(G_l) \cup \ldots \cup V(G_l)$ and $E(G) = E(G_1) \cup E(G_l) \cup \ldots \cup E(G_l) \cup \{(x,y) \mid x \in V(G_i), y \in V(G_j) : (i,j) \in G_0\}$. It is straightforward to see that if $M_i = ind(G_i)$ then $ind(G) = \mathcal{M}_0(\mathcal{M}_1, \ldots, \mathcal{M}_l)$.

In the following corollary of Theorem 5.1 we prove a lower bound of $n^{0.226}$ on the competitive ratio of on-line algorithms for the maximum independent set of a graph. Our lower bound is better than the one in Theorem 5 of [3] but worse than the one in Theorem 6 of [3]. The proof of Theorem 5 in [3] contains a fixable error. In the appendix of [7] we describe the error and give a direct way to fix it. Alternative one can obtain this theorem also as corollary of our Theorem 5.1. Bartal et al. withdrew their claim for a stronger bound stated in their Theorem 6. Thus the bound in Corollary 5.2 is currently the best known.

COROLLARY 5.2. *There exists a graph G such that $e(ind(G)) \le 1/n^{0.226}$, or in other words the competitive ratio of any online algorithm for the maximum independent set in G is at least $n^{0.226}$.*

Proof. Let H be the graph obtained from $K_{3,3}$ by deleting a perfect matching. Let $\{1,3,5\}$ be the vertices on one side of the bipartition and $\{2,4,6\}$ the vertices on the other side and $\{(1,2),(3,4),(5,6)\}$ the matching deleted. Let P be a probability distribution on $S(B)$ such that $P(1352) = 1/3$, $P(1354) = 1/3$, $P(1356) = 1/3$, and on every other sequence P is zero. One can calculate using this distribution that $e'(ind(H)) \le 2/3$. Let G be the graph obtained by taking the product of H with disjoint copies of itself $\log_6 n$ times. By applying Corollary 5.1 to $ind(G)$ the lemma follows.

We can also use Theorem 5 to prove the upper bound of $O(1/(\log k \log l))$ on $e(k,l)$ given by Awerbuch et al. [1]. We formulate this result precisely in as the following theorem.

THEOREM 5.2. *For $k \ge \log l$ and $l \ge \log k$, and $k,l \ge 4$, $e(k,l) \le 8/[(1+\log k - \log\log l)(1+\log l - \log\log k)]$. For $k \le \log l$ or $l \le \log k$, $e(k,l) \le 8/\min(k,l)$. (In this theorem all* logs *are based 2)*

6 Greedy algorithms and Extensions of Matroids

Although the focus of our study is randomized algorithms, we have examined subclasses of the former as well, such as deterministic algorithms and variants of greedy algorithms.

Recall that a *matroid* is a monotone set system $\mathcal{M}$ with the property that for every $M_1, M_2 \in \mathcal{M}$, $|M_1| < |M_2|$ it is possible to find an element $x \in M_2 \setminus M_1$ such that $M_1 \cup \{x\} \in \mathcal{M}$ (for more information about matroid theory see [14]). Our starting point for studing deterministic algorithms was the following simple theorem.

THEOREM 6.1. *e(M)=1 if and only if $\mathcal{M}$ is a matroid.*

We have found a subclass of the deterministic algorithms such that we can fully characterize set systems for which there exists an algorithm with performance at least c from this class.

Let $\mathcal{M}$ be a monotone set system, and $\mathcal{M}' \subseteq \mathcal{M}$ be a monotone sub-system. The *generalized greedy algorithm* for $\mathcal{M}$ with respect to $\mathcal{M}'$, denoted by $greedy_{\mathcal{M}'}$, picks an element x from an input sequence s iff x together with the set of already picked elements belongs to $\mathcal{M}'$. The following theorem characterizes set systems for which there exists a generalized greedy algorithm with performance at least c.

THEOREM 6.2. *Let $\mathcal{M}$ be a monotone set-system. Then there exists a generalized greedy algorithm for $\mathcal{M}$ with performance at least $c(\le 1)$ iff there exists a monotone set system $\mathcal{M}' \subseteq \mathcal{M}$ such that for every $M' \in \mathcal{M}'$ and for every $M \in \mathcal{M}$ with $c|M| > |M'|$ there exists an $x \in M \setminus M'$ such that $\{x\} \cup M \in \mathcal{M}'$.*

During the study of specific systems we observed that even if the set system $\mathcal{M}$ is not a matroid, there are elements of $B = B(\mathcal{M})$ that it is always good to pick greedily. Interestingly we have found a purely set theoretical characterization of such elements.

DEFINITION 6.1. *An element $x \in B$ is* greedy *for $\mathcal{M}$ if for every $M_2 \subset M_1 \in \mathcal{M}$ such that $M_2 \cup \{x\} \in \mathcal{M}$, there is an $M' \in \mathcal{M}$ such that $M_2 \cup \{x\} \subseteq M' \subset M_1 \cup \{x\}$, and $|M'| = |M_1|$.*

Example: Let $B = \{1,2,3,4\}$, $\mathcal{M} = \{\{1,2\},\{1,3,4\},\{2,3,4\}\}^c$. Then 3 and 4 are greedy for $\mathcal{M}$.

The following theorem describes a way to change an algorithm to another one which works in a greedy manner on a prescribed set of greedy elements, and has the same or better performance.

THEOREM 6.3. *Let G be a set of greedy elements for $\mathcal{M}$. Then for any randomized algorithm A there is a randomized algorithm A' such that*

1. *When A' encounters an element $x \in G$ then it picks x with probability one if x and the elements already picked by A' form an independent set.*
2. *For every input sequence s, $e(A,s) \le e(A',s)$.*

Theorem 6.3 is particularly useful when one comes to design an algorithm for a set system $\mathcal{M}$ for which the greedy elements are known or can be efficiently computed. We finish this section by remarking that every element $x \in B$ is greedy for $\mathcal{M}$ if and only if $\mathcal{M}$ is a matroid.

7 Discussion

We have defined a model for online computation on monotone set systems in which the decision of the player is binary. The model we define has nice mathematical properties and it generalizes the model of [3] in which the set systems arise from hereditary graph properties. It also captures a basic special case of the Video On Demand problem [2, 1]. We proved theorems that bound the performance of set systems produced from other systems through set theoretic operation such as product and union. We also analyzed the performance of specific interesting set systems such as $\mathcal{M}_{k,3}$.

Many intriguing open question are left open. One set of questions is related to systems with disjoint maximal sets (Section 3). Here are few of them. (1) Is our algorithm for $\mathcal{M}_{k,3}$ is indeed the best possible? (2) Prove or disprove Conjecture 3.1. (3) Can one generalize our algorithm for $\mathcal{M}_{k,3}$ to $\mathcal{M}_{k,4}$ or even to $\mathcal{M}_{k,l}$? (4) Find a precise formula for $e(k,l)$.

Other questions are about the model in general. Here are few examples: (1) For an arbitrary set system we know that $1/\sqrt{2n} \leq e(\mathcal{M})$ ([3][Corollary 19] and Theorem 4.1), on the other hand we do not know of a system with matching performance, The worse system we know of is the one of Corollary 5.2 for which $e(\mathcal{M}) \leq n^{-0.226}$. Is it possible to close this gap between the upper and the lower bounds? (2) Find a set theoretic characterization of those set systems with performance $\Omega(1)$, or/and of those set systems with performance $\Omega(1/polylog(n))$. (3) Find a set theoretic characterization for those systems for which there exists a deterministic algorithm with performance $\Omega(1)$.

Acknowledgments: We acknowledge Joel Spencer for his contribution to Theorem 3.4, David Johnson for his comments on an early draft of this paper, and Anna Karlin for pointing us to reference [1].

References

[1] B. Awerbuch, Y. Azar, A. Fiat, and T. Leighton. Making commitments in the face of uncertainity: How to pick a winner almost every time. In *Proc. 28th ACM Symposium on Theory of Computing*, pages 519–530, 1996.

[2] A. Bar-Noy, J. A. Garay, A. Herzberg, and S. Aggarwal. Sharing video on demand. Technical report, IBM Technical Memorandum, 1996.

[3] Y. Bartal, A. Fiat, and S. Leonardi. Lower bounds for on-line graph problems with application to on-line circuit and optical routing. In *Proc. 28th ACM Symposium on Theory of Computing*, pages 531–540, 1996.

[4] A. Borodin, N. Linial, and M. E. Saks. An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4):745–763, 1992.

[5] A. Fiat, R. Karp, M. Luby, L. McGeoch, D. D. Sleator, and N. Young. Competitive paging algorithms. *J. Algorithms*, 12:685–699, 1991.

[6] M. M. Hallsórsson. Parallel and online graph coloring. Technical report, Science Institute, University of Iceland. To appear in *J. Algorithms*.

[7] H. Kaplan and M. Szegedy. Online Complexity of Monotone Set Systems. http://www.research.att.com/~hkl/papers/online.ps

[8] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proc. 22nd ACM Symposium on Theory of Computing*, pages 352–358, 1990.

[9] E. Koutsoupias and C. H. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5):971–983, 1995.

[10] S. Leonardi. Private communication, April 1998.

[11] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for on-line problems. *J. Algorithms*, 11(2):208–230, 1990.

[12] N. Reingold, J. Westbrook, and D. D. Sleator. Randomized algorithms for the list update problem. *Algorithmica*, 11(1):15–32, 1994.

[13] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.

[14] D. J. A. Welsh. *Matroid Theory*. Academic Press, London, 1976.

[15] A. C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. 17th Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.