**Dagstuhl Seminar on**

# Competitive Algorithms

June 20–25 1999

Organized by:

Amos Fiat (Tel-Aviv University)
Anna Karlin (University of Washington)
Gerhard Woeginger (TU Graz)

# Summary

Decision making can be considered in two different contexts: making decisions with complete information, and making decisions based on partial information. A major reason for the study of algorithms is to try to answer the question: 'Which is the better algorithm?' The study of the computational complexity of algorithms is useful for distinguishing the quality of algorithms based on the computational resources used and the quality of the solution they compute. However, the computational complexity of algorithms may be irrelevant or a secondary issue when dealing with algorithms that operate in a state of uncertainty. 'Competitive' analysis of algorithms has been developed in the study of such algorithms.

Competitive analysis is useful in the analysis of systems that have some notion of a time progression, that have an environment, that respond in some way to changes in the environment, and that have a memory state. Competitive analysis is used for so-called 'on-line algorithms' that have to respond to events over time. Competitive analysis is used whenever the nature of the problem is such that decisions have to be made with incomplete information.

The Dagstuhl meeting on *Competitive Algorithms* brought together researchers with affiliations in Austria, Canada, Czech Republic, Denmark, Germany, Hungary, Israel, Italy, the Netherlands, Poland, and the USA. 36 presentations were given. The abstracts of most of these presentations are contained in this seminar report. Moreover, there is a list with some open problems that were mentioned in the open problem session on Wednesday evening.

As always, we enjoyed the social aspects of a Dagstuhl meeting. As always, we had a wine-and-cheese party every evening. And as always, we are looking forward to the next Dagstuhl workshop.

Amos Fiat
Anna Karlin
Gerhard Woeginger

# Abstracts

### Page Replacement for General Caching Problems

Susanne Albers

Caching (paging) is a well-studied problem in online algorithms, usually studied under the assumption that all pages have a uniform size and a uniform fault cost (*uniform caching*). However, recent applications related to the web involve situations in which pages can be of different sizes and costs. This *general caching problem* seems more intricate than the uniform version. In particular, the offline case itself is NP-hard. In this talk we develop good offline page replacement policies for the general caching problem, with the hope that any insight gained here may lead to good online algorithms. Our main result is that by using only a small amount of additional memory, say $O(1)$ times the largest page size, we can obtain an $O(1)$-approximation to the general caching problem. Note that the largest page size is typically a very small fraction of the total cache size, say 1%. Our result uses a new rounding technique for linear programs which may be of independent interest. We also present a randomized online algorithm for the Bit Model which achieves a competitive ratio of $O(\ln(1 + 1/c))$ while using $M(1 + c)$ memory.

This is joint work with Sanjeev Arora and Sajeev Khanna.

### The adversarial bandit problem

Peter Auer

In the multi-armed bandit problem, a gambler must decide which arm of $K$ non-identical slot machines to play in a sequence of trials so as to maximize his reward. This classical problem has received much attention because of the simple model it provides of the trade-off between exploration (trying out each arm to find the best one) and exploitation (playing the arm believed to give the best payoff). Past solutions for the bandit problem have almost always relied on assumptions about the statistics of the slot machines.

In this work, we make no statistical assumptions whatsoever about the nature of the process generating the payoffs of the slot machines. We give a solution to the bandit problem in which an adversary, rather than a well-behaved stochastic process, has complete control over the payoffs. In a sequence of $T$ plays, we prove that the expected per-round payoff of our algorithm approaches that of the best arm at the rate $O(T^{-1/2})$. In addition, we consider a setting in which the player has a team of "experts" advising him on which arm to play; here, we give a strategy that will guarantee expected payoff close to that of the best expert. We apply our result to the problem of learning to play an

unknown repeated matrix game. A more involved application of our techniques gives an on-line algorithm which adaptively selects a path with (almost) minimum average weight in a graph where the edge weights change over time.

This is joint work with N. Cesa-Bianchi, Y. Freund, and R. Schapire

**Independent sets in hypergraphs with applications to routing via fixed paths**

YOSSI AZAR

The problem of finding a large independent set in a hypergraph by an online algorithm is considered. We provide bounds for the best possible performance ratio of deterministic vs. randomized and non-preemptive vs. preemptive algorithms. Applying these results we prove bounds for the performance of online algorithms for routing problems via fixed paths over networks.

Joint work with N. Alon and U. Arad.

**The accommodating ratio: a measure for the performance of on-line algorithms**

JOAN BOYAR

The Unit Price Bin Packing Problem was used as an example to motivate the definition of the *accommodating ratio*. In this variant of bin packing, there is a fixed number $n$ of bins, each of size $k$, and all requests have integer sizes. There is a requirement that the algorithms considered be fair, i.e., if when a request is given there is room for it in some bin, then it must be accepted. The goal is to maximize the number of objects accepted.

The accommodating ratio is very similar to the competitive ratio in that it is the worst case ratio of the on-line algorithm's performance to the optimal off-line algorithm's performance. The difference is that this worst case is not over all possible sequences, but rather over all sequences which an optimal off-line algorithm could have accommodated. The accommodating ratio was originally defined for the Seat Reservation Problem.

For the Unit Price Bin Packing Problem, Worst-Fit has a better competitive ratio than First-Fit, but First-Fit has a better accommodating ratio than Worst-Fit. Thus, the accommodating ratio can be useful in distinguishing between two algorithms.

This work is joint with Kim Skak Larsen and Morten Nyhave Nielsen.

**On-Line prediction with experts: recent results**

NICOLÒ CESA-BIANCHI

We present sequential strategies for assigning probabilities to the elements that may appear next in a sequence of data. The goal is to minimize the regret under log loss over the worst possible sequence. That is, to minimize the worst-case drop in the log-likelihood

of the final sequence when measured under the assigned probabilities, as opposed to being measured under the best assignment in a given class of strategies (or experts). Using tools from empirical process theory, we prove a general upper bound on the best possible (minimax) regret that depends on the metric properties of the class of experts. This extends previous results by Opper and Haussler. Finally, we connect the minimax regret under log loss with a corresponding minimax quantity related to the theory of portfolio selection introduced by Cover.

Joint work with Gábor Lugosi.

**The 3-server problem in the plane**
Marek Chrobak

In the $k$-server problem we wish to minimize, in an online fashion, the movement cost of $k$ servers in response to a sequence of requests (we assume that $k \geq 2$). The request issued at each step is specified by a point $r$ in a given metric space $M$. To serve this request, one of the $k$ servers must move to $r$.

It is known that if $M$ has at least $k+1$ points then no online algorithm for the $k$-server problem in $M$ has competitive ratio smaller than $k$. The best known upper bound on the competitive ratio in arbitrary metric spaces, by Koutsoupias and Papadimitriou, is $2k-1$. There are only a few special cases for which $k$-competitive algorithms are known: for $k = 2$, when $M$ is a tree, or when $M$ has at most $k + 2$ points.

We prove that the Work Function Algorithm is 3-competitive for the 3-Server Problem in the Manhattan plane. As a corollary, we obtain a 4.243-competitive algorithm for 3 servers in the Euclidean plane. The best previously known competitive ratio for 3 servers in these spaces was 5.

This is joint work with Wolfgang W. Bein and Lawrence L. Larmore

**A new algorithm for bin packing**
János Csirik

In the classical bin packing problem, a list of items and a bin size $B$ is given. We have to pack the items into a minimum number of bins such that the sum of items in each bin is $B$ or less. We will assume that $B$ and the item sizes are integers. Let $P$ be a packing of list $L$ and let $N_P(h)$ be the number of bins whose contents have total size equal to $h$. Let the sum of squares $ss(P)$ be $\sum_{h=1}^{B-1} N_P(h)$.

The Sum-of-Squares Algorithm is the following: let $a$ be the next item to be packed. A legal bin for $a$ is either empty or has a current empty space which is enough for $a$. Place $a$ into a legal bin so as to yield the minimum possible value for the resulting $ss(P')$.

We gave results for packing lists from uniform discrete distribution. We have a great number of experiments showing that this new online algorithm has a better average behaviour than the well-known best fit algorithm.

**Scheduling with precedence constraints**
LEAH EPSTEIN

We consider several variants of scheduling jobs on $m$ parallel machines. We discuss deterministic and randomized algorithms, preemptive or non-preemptive, independent jobs or precedence constraints and known or unknown durations. We also consider all four machine models which are identical machines, uniformly related machines, restricted assignment, and unrelated machines. We state results on on-line scheduling in some variants, and give some new results. We give upper and lower bounds of $O(m)$ and $\Omega(m)$ for deterministic and randomized, preemptive and non-preemptive scheduling with precedence constraints, with known or unknown durations for the *restricted assignment* model. If only consistent precedence constraints are allowed, we give lower and upper bounds of $\Omega(\log m)$ and $O(\log m)$ for the same problems. We also consider the same variants on *related machines*. We give a lower bound of $\Omega(\sqrt{m})$ which is tight.

Joint work with Yossi Azar.

**On Some Maximum Weight Independent Set Problems**
THOMAS ERLEBACH

Given an undirected graph $G = (V, E)$ with positive node weights $w(v)$, the maximum weight independent set problem is to compute a subset $A \subseteq V$ such that $A$ is an independent set (no two nodes in $A$ are adjacent) and such that $w(A) := \sum_{v \in A} w(v)$ is maximized. We consider a simple algorithm, called GREEDY$_\alpha$, that processes the nodes in some order and selects each node $v$ if $w(C_v) \leq \alpha \cdot w(v)$, where $C_v$ is the set of currently selected nodes that are adjacent to $v$. If $v$ is selected, the nodes in $C_v$ are discarded. In the end, the selected nodes are returned. Here, $\alpha$ is a parameter that can be chosen between 0 and 1. For $\alpha = \frac{1}{2}$, GREEDY$_\alpha$ is known as the "double-the-gain" strategy.

We discuss two applications of GREEDY$_\alpha$. The first is a job interval selection problem (JISP), where the input is a set of jobs, each of which consists of a certain number of intervals. The goal is to compute a subset of disjoint intervals with at most one interval from each job such that the total weight of the intervals in the subset is maximized. If we let GREEDY$_\alpha$ process the intervals in order of increasing right endpoints, it achieves approximation ratio $3 + 2\sqrt{2} \approx 5.83$ for $\alpha = \sqrt{2} - 1$ in the case where all intervals of a job must have the same weight, and approximation ratio 8 for $\alpha = \frac{1}{2}$ if intervals can have arbitrary weights. For the former case, we prove a competitive lower bound of $2 + 2\sqrt{2} \approx 4.83$ on the approximation ratio of any deterministic algorithm that processes the intervals in the same order as GREEDY$_\alpha$ and that must select or reject each interval without knowledge of future requests. The second application is the weighted maximum edge-disjoint paths problem in bidirected trees, where GREEDY$_{\frac{1}{2}}$ achieves approximation ratio 8 if the paths are processed in order of non-increasing distance from the root of the tree.

In general, we show that GREEDY$_{\frac{1}{2}}$ achieves approximation ratio $4k$ if the nodes of

a graph $G$ are processed in an order $v_1, \ldots, v_n$ such that the subgraph of $G$ induced by $\Gamma(v_i) \cap \{v_{i+1}, \ldots, v_n\}$ has independent set size at most $k$.
Note: The 5.83-approximation for JISP with equal weight for intervals of the same job was already found in a different context by Bar-Noy et al. [1]. They also present an LP-based approach that can easily be adapted to JISP with arbitrary weights and gives a 2-approximation (our lower bound does not apply to this algorithm).

Joint work with Frits Spieksma.

[1] A. Bar-Noy, S. Guha, J. Naor, B. Schieber. Approximating the Throughput of Multiple Machines under Real-Time Scheduling. Presented at this seminar, published in *Proceedings of STOC'99*, pp. 622–631.

**Dynamic traitor tracing**
Amos Fiat

We study the problem of how to watermark content so as to distribute different versions of watermarked content (such as a movie) to different subscribers. We consider an online variant of this problem where we assume that pirate broadcasts can be viewed and subsequent transmissions can be modified based upon the current pirate behaviour. This problem translates into the following online problem:

Our goal is to isolate "bad users" until we have certain proof that such a user is bad, at which point the user can be removed. Initially, we are unaware of any "bad users", and transmit the identical movie to all subscribers, when a pirate transmission is intercepted then we know that there is at least one "bad user". At any point in time we have a lower bound on the number of "bad users", $k$, (initially zero) and may partition the user group into up to $k+1$ different sets, each of which is given a different version of a scene from the movie. Let us call one such tranmission an experiment. The goal is to isolate all bad users in singleton sets and to minimize the number of experiments required to do so.

We found an algorithm to do so that requires $3^p \cdot p \log n$ experiments, where $p$ is the total number of bad users. An open problem posed in the open problems session was to improve the above bound. Jiri Sgall found a much better algorithm requiring only $p^3 \log n$ experiments.

Joint work with Tamir Tassa.

**On the polygon exploration problem**
Frank Hoffmann

We present an online strategy that enables a mobile robot with a $2\pi$–vision system to explore an unknown simple polygon. The resulting length of a tour created by this strategy is at most 26.5 as long as the shortest watchman tour that could be computed off–line (knowing the polygon).

Especially, we show how the analysis of our strategy leads to a novel geometric struc-

ture called the *angle hull*. Let $D$ be a connected and relative convex region inside a simple polygon $P$. The angle hull of $D$, $AH(D)$, is defined to be the set of all points in $P$ that can see two points of $D$ at a right angle. We show that for any $P$ and $D$ the length of the perimeter of $AH(D)$ is at most twice as long as the perimeter of $D$, and, moreover, this bound is tight.

Finally, we discuss open problems related to the question how to explore polygons with a certain number of holes.

Joint work with Rolf Klein, Christian Icking, and Klaus Kriegel

**Linear Algebra and Web Search**
RAVI KANNAN

With the abundance of information (especially on the web), an interesting problem is: Given a large collection of documents, find correlations bewteen the documents and classify the collection into clusters of similar documents. Such clustering problems arise in other application areas as well. Linear Algebra based methods have long been proposed for finding such correlations; but traditional Linear Algebra takes too long to be used in many modern applications.

We present results which prove that in many contexts, it suffices to do Linear Algebra on a randomly chosen submatrix of the entire matrix of data. Using the resulting algorithms, we have implemented a document classification system which can be used to classify the results of a keyword search of the web very fast; thus making the use of such correlation analysis a feasible addition to a web search engine.

**Game coloring number**
HAL KIERSTEAD

Let $G = (V, E)$ be a finite graph and $\Pi(G)$ be the set of linear orders on $V$. Each order $L \in \Pi(G)$ determines an orientation $G_L = (V, E_L)$ of $G$ defined by $E_L = \{(v, u) : \{v, u\} \in E \text{ and } v > u \text{ in } L\}$. Let $\Delta^+(G_L)$ denote the maximum outdegree of $G_L$. The *coloring number* of $G = (V, E)$ is defined by

$$\operatorname{col}(G) = 1 + \min_{L \in \Pi(G)} \Delta^+(G_L).$$

Clearly the chromatic number $\chi(G) \leq \operatorname{col}(G)$, since if $\operatorname{col}(G) = 1 + \Delta^+(G_L)$, then First-Fit applied to the vertices of $G$ in the order $L$ will use at most $\operatorname{col}(G)$ colors. We shall consider competitive versions of the graph parameters *chromatic number* and *coloring number*.

The *chromatic game* is played on $G$, using a set of colors $C$, by two players Alice and Bob with Alice playing first. The players take turns choosing and properly coloring uncolored vertices of $G$ with colors from $C$. Bob wins if at some time one of the players has no legal move; otherwise Alice wins when the players eventually create a proper

coloring of $G$. The *game chromatic number* of $G$, denoted $\chi_g(G)$, is the least $t$ such that Alice has a winning strategy when $|C| = t$.

The *ordering game* is played on $G$ with a target $t$ by Alice and Bob with Alice playing first. In this game the players take turns choosing unchosen vertices. This creates a linear ordering $L$ on the vertex set of $G$ with $x < y$ iff $x$ is chosen before $y$. The *score* of the game is $\Delta^+(G_L) + 1$. Alice wins if the score is at most $t$; otherwise Bob wins. The *game coloring number* $\mathrm{col}_g(G)$ of $G$ is the least cardinal $t$ such that Alice has a winning strategy for the ordering game played on $G$ with target $t$. The game coloring number was originally introduced as a tool for studying the game chromatic number, but in retrospect it seems that it may actually be a more interesting parameter.

We consider the problem of bounding the game coloring number, and therefore the game chromatic number, of various classes of graphs, including forests, subgraphs of chordal graphs, outerplanar graphs, planar graphs, and line graphs. In all these cases the best known bounds on the game chromatic number are obtained by bounding the game coloring number. Here we present a single simple strategy for Alice to use to play the ordering game. This strategy is based on a preordering of the vertices of the graph $G$ on which the game is played. We define a rank function on such preorderings and then bound the score of the game in terms of the rank of the preordering. This approach allows us to very easily obtain all known results for these cases by showing that the various graphs all have preorderings with small enough rank. We also obtain the new result that if $G$ is embeddable on an orientable surface of genus $g$ then $\mathrm{col}_g(G) \leq \frac{3\sqrt{73+96g}+41}{4} = (1 + o(1))\sqrt{54g}$.

**Online two-machine preemptive job shop scheduling**
TRACY KIMBREL

We consider online and offline algorithms for special cases of preemptive job shop scheduling to minimize makespan. These special cases are of interest because they commonly arise in the scheduling of computer systems. Our results are

- A randomized online algorithm for the two-machine preemptive job shop that is 1.5-competitive against oblivious adversaries;
- Lower bounds showing that the randomized bound of 1.5 and the trivial deterministic upper bound of 2 are asymptotically tight;
- A linear-time offline 1.5-approximation algorithm for the two-machine preemptive job shop.

Interestingly, the randomized algorithm requires only a single random bit.

Joint work with Jared Saia.

**On the work function algorithm**
ELIAS KOUTSOUPIAS

We present a new "pictorial" proof that the WFA has competitive ratio $2k-1$ for the $k$-server problem. The proof technique is applicable also to the generalization of the $k$-server problem when the off-line algorithm has fewer than $k$ servers. We give two upper bounds of the cost $\text{WFA}(\rho)$ of the Work Function Algorithm. The first upper bound is $k\text{OPT}_h(\rho) + (h-1)\text{OPT}_k(\rho)$, where $\text{OPT}_m(\rho)$ denotes the optimal cost to service $\rho$ by $m$ servers. The second upper bound is $2h\text{OPT}_h(\rho) - \text{OPT}_k(\rho)$ for $h \leq k$. Both bounds imply that the Work Function Algorithm is $(2k-1)$-competitive against an adversary with $k$ servers. The proofs are simple and intuitive and they do not involve potential functions.

**Online dial-a-ride problems: competitive analysis and reasonable load**
SVEN O. KRUMKE

We present results for online "dial-a-ride" transportation problems: Objects are to be transported between the vertices of a given graph. Transportation requests arrive online, specifying the objects to be transported and the corresponding source and target vertex. These requests are to be handled by a server which starts its work at a designated origin vertex and which picks up and drops objects at their starts and destinations. The server can move at constant unit speed. After the end of its service the server returns to its start. The goal of the basic problem is to come up with a transportation schedule for the server which finishes as early as possible, i.e., which minimizes the makespan.

We show a lower bound of $1 + \sqrt{2}/2 \approx 1.70$ for the competitive ratio of any deterministic algorithm which holds even if the graph is a simple path. We then analyze two simple and natural strategies which we call REPLAN and IGNORE. REPLAN completely discards its schedule and recomputes a new one when a new request arrives. IGNORE always runs a (locally optimal) schedule for a set of known requests and ignores all new requests until this schedule is completed.

We show that both strategies, REPLAN and IGNORE are 5/2-competitive.

We then analyze the algorithms with request sets that fulfill a certain worst-case restriction: roughly speaking, a set of requests for the online dial-a-ride problem is "reasonable" if the requests that come up in a sufficiently large time period can be served in a time period of at most the same length. This new notion is a stability criterion implying that the system is not overloaded.

The new concept is used to analyze the online dial-a-ride problem for the minimization of the maximal resp. average flow time. Under reasonable load it is possible to distinguish the performance of REPLAN and IGNORE for this problem, which seems to be impossible by means of classical competitive analysis. In particular, under reasonable load we are able to establish upper bounds on the maximum flow time for the IGNORE strategy.

Joint work with Jörg Rambau and Dietrich Hauptmeier.

**Minimizing the Flow Time without Migration**
Stefano Leonardi

We consider the classical problem of scheduling jobs in a multiprocessor setting in order to minimize the flow time (total time in the system). The performance of the algorithm, both in offline and online settings, can be significantly improved if we allow preemption: i.e., interrupt a job and later continue its execution, perhaps migrating it to a different machine. Preemption is inherent to make a scheduling algorithm efficient. While in case of a single processor, most operating systems can easily handle preemptions, migrating a job to a different machine results in a huge overhead. Thus, it is not commonly used in most multiprocessor operating systems.

The natural question is whether migration is an inherent component for an efficient scheduling algorithm, in either online or offline setting.

Leonardi and Raz (STOC'97) showed that the well known algorithm, shortest remaining processing time (SRPT), performs within a logarithmic factor of the optimal algorithm. Note that SRPT must use both preemption and migration to schedule the jobs. It is also known that in the on-line setting that no algorithm can achieve a better bound.

In this work it is presented a new algorithm that does not use migration, works online, and is just as effective (in terms of competitive ratio) as the best known algorithm (SRPT) that uses migration.

(Joint work with B. Awerbuch, Y. Azar and O. Regev)

**On-line algorithms for the network RAM problem**
Ketsiya E. Meirman

The network RAM problem can be describe as follows: given a computing cluster, with several nodes that are connected by a fast LAN. Suppose that a process needs to access a large data segment, which does not fit in the main memory of any node. Assume that the data segment is placed in the main memory of several nodes. The motivation is that remote paging, in which a remote page is brought to the local memory through a fast LAN is much faster than reading a page from a local disk. Assume further that the operating system of all the nodes supports preemptive process migration, i.e., a process can stop running in one node and after a migration, can resume its execution in another node. In the cluster environment, the execution of the process involves access to local pages (in the memory of the current node) as well as access to remote pages, in the memory of other nodes. If the number of remote paging becomes excessive, then it might be beneficial to migrate the process to the remote node rather than to perform remote paging. Since the list of future access pages is not known, the problem is to design an algorithm that decides on-line if to page or to migrate, to minimize the execution time of the process. This problem assume a static page model. In the static memory model a page loaded onto node's memory can not move to another one. In each node there is one

special memory page that holds temporarily remote pages. On the dynamic page model where the whole memory can be reorginized the problem is MAX-SNP hard.

The above, network RAM problem, has two cost models. In the partial cost model, access to local pages costs zero, access to a remote page costs one, while process migration costs a constant D, proportional to the process size (in pages). For this model, the network RAM problem maps to the file migration problem, for which it was proved in [4,3] that there is a deterministic lower bound of 3. We note that Black & Sleator [3] and Bartal et al. [2,1] developed algorithms, M and RFWF respectively, which achieve this lower bound.

In our work we developed a new, 3.56 competitive algorithm, called cD-FRE. Experimental results shows that for many real cases, e.g., simulation of matrix multiplications, sorts, etc., algorithm cD-FRE performs better than algorithm M and RFWF. To explain this we analyze the problem in the full cost model, in which a local access to a page costs 1, remote access costs some constant $s$ and process migration costs $sD$. In the full paper we present definitions of locality of reference for this model, and we show that cD-FRE exploits locality of reference, while both M and RFWF does not. Our work was inspired by that of Torng [5] on the locality of reference in the paging problem. In the paper we also present a new algorithm, called M-FRE, which is a combination of M and FRE. This algorithm outperforms all the previous algorithms.

Joint work with Amnon Barak.

[1] Bartal Y., Distributed Paging. In Online Algorithms, Fiat, A.; Woeginger G.J.; Lecture Notes in Computer Science, Vol 1442, Springer, May 1998.

[2] Bartal, Y.; Charikar, M.; Indyk P., On Page Migration and Othe Relaxed Task Systems. In Proc. of the 8th Ann. ACM-SIAM Symp. on Discrete Algorithms, pages 43-52, January 1997.

[3] Black, D.L. and Sleator D.D., Competitive Algorithms for Replication and Migration Problems. In Technical report CMU-CS-89-201, Department of Computer Science, Carnegie Mellon University, November 1989.

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

[5] Torng, E., A Unified Analysis of Paging and Caching. In Algorithmica, 20:175-200, 1998.

### Refined Locality of Reference for Paging

MANOR MENDEL

In this talk we refine the locality of reference concept, captured by the *access graph model* of [1], to allow temporal changes in the behavior of the underlying process. We formalize this by introducing the concept of an *extended access graph*. We then give truly

online algorithms for extended access graphs with "low rate of change" in the locality of reference, and show almost tight impossibility results for graphs with "high rate of change" in the locality of reference.

Joint work with Amos Fiat.

[1] Allan Borodin, Sandy Irani, Prabhakar Raghavan, and Baruch Schieber. Competitive paging with locality of reference. *J. Comp. Syst. Sci.*, 50(2):244–258, April 1995.

**The accommodating function**
Morten Nyhave Nielsen

The competitive ratio as a measure for the quality of on-line algorithms has been criticized for giving bounds that are unrealistically pessimistic and for not being able to distinguish between algorithms with very different behavior in practical applications.

A new measure, the *accommodating function*, for the quality of on-line algorithms is presented. The accommodating function is a generalization of both the competitive ratio and the accommodating ratio. As an example, we investigate the measure for a variant of bin-packing in which the goal is to maximize the number of objects put in $n$ bins.

We give upper and lower bounds for the accommodating function for any fair algorithm for this problem, and tighter bounds for the concrete algorithm, First-Fit.

Joint work with Joan Boyar and Kim S. Larsen.

**Caching for web searching**
John Noga

We discuss web caching when the input sequence is a depth first search traversal of some tree. There are at least two good motivations for investigating tree traversal as a search technique on the web: First, empirical studies of people browsing and searching the WWW have shown that user access patterns commonly are nearly depth first traversals of some tree. Secondly, the problem of visiting all the pages on some web site using anchor clicks (clicks on links) and back button clicks — by far the two most common user actions — reduces to the problem of how to best cache a tree traversal sequence.

We show that for tree traversal sequences the optimal offline strategy can be computed efficiently. In the bit model, where the access time of a page is proportional to its size, we show that the online algorithm LRU is 2-competitive against an adversary with an **unbounded** cache as long as LRU has a cache large enough to store the largest **two** items in the input sequence. In the general model, where pages have arbitrary access times and sizes, we show that in order to be constant competitive, any online algorithm needs a cache large enough to store $\Omega(\log n)$ pages; here $n$ is the number of distinct pages in the input sequence. We provide a matching upper bound by showing that the online algorithm Landlord is constant competitive against an adversary with an unbounded cache if Landlord has a cache large enough to store the $\Omega(\log n)$ largest pages. This is

further theoretical evidence that Landlord is the "right" algorithm for web caching.

Joint work with Bala Kalyanasundaram, Kirk Pruhs, and Gerhard Woeginger

**Computer-aided complexity classification of dial-a-ride problems**
WILLEM DE PAEPE

A Dial-a-Ride problem is a problem in which a set of servers with a given capacity and speed has to serve a set of rides. Every ride $i$ is characterized by a source $s_i$, a destination $t_i$, a release time $r_i$, and a deadline $d_i$. A server has to pick up an item at time $x \geq r_i$ in $s_i$ and deliver it at time $x' \leq d_i$ in $t_i$. We assume that all items are identical. Preemption may be allowed, and there might be precedence constraints on the order in which the rides are served. The goal is to minimize a certain objective function. Dial-a-Ride problems are defined on some metric space in which an origin $O$ is specified. The servers are supposed to start and end in $O$. We assume the number of rides is finite for every instance of a Dial-a-Ride problem. Dial-a-Ride problems can be studied from an offline as well as an online point of view.

In [2] Lawler, Lenstra, Rinnooy Kan and Shmoys present a classification for scheduling problems that enables us to give a short and unambiguous notation for most scheduling problems. We propose a similar classification for Dial-a-Ride problems and we describe the set of problems that can be classified in this way. Whereas a scheduling problem is represented by the contents of three fields, we need four fields to describe a Dial-a-Ride problem. In these fields information about the servers, the rides, the metric space and the objective are given, respectively. Any combination of entries can be interpreted as a Dial-a-Ride problem and the problem class under study consists of all possible combinations of entries. The problem class consists of 9160 problems.

Using the relations between the entries of each field, we define a partial ordering " $\rightarrow$ " on the problems in the problem class that is useful in studying the structure of the problem class. A similar approach for scheduling problems can be found in [1]. We show how this partial ordering can be helpful in determining the computational complexity of the problems in the problem class, without studying every single problem separately. To this end we use the computer program DARCLASS which is able to determine whether or not for any two problems $P$ and $P\prime$ the relation $P \rightarrow P\prime$ holds. At the moment we have determined the complexity of 8457 problems using only 55 NP-completeness results and 33 polynomial algoritmes. It turns out that more than 97 percent of these problems are NP-complete.

Joint work with Jan Karel Lenstra, Leen Stougie.

[1] B.J. Lageweg & J.K. Lenstra, E.L. Lawler, A.H.G. Rinnooy Kan, Computer-Aided Complexity Classification of Combinatorial Problems, Communications of the ACM, November 1982, Volume 25, Number 11, 817-822.

[2] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, Sequencing and Scheduling: Algorithms and Complexity, Handbooks in OR & MS (1993), 445-522.

**A quick tour of resource augmentation results**
Kirk Pruhs

In the context of approximating NP-hard optimization problems, and in the context of online problems, one compares the performance of a limited algorithm (one that can only run in polynomial time, and one that has no knowledge of the future, respectively) against a omnipotent clairvoyant adversary. To be more concrete consider a minimization problem, and let $L$ be the limited algorithm. Then the competitive ratio, or performance ratio, or ratio bound, or worst-case ratio, etc., of $L$ is then

$$\rho(n) = \max_I \frac{L(I)}{A(I)}$$

where $L(I)$ denotes the cost of the solution produced by the algorithm $L$ on input $I$, $A(I)$ denotes the cost of the optimal solution, and the maximum is over all inputs $I$ of size $n$. The competitive ratio for the problem is then essentially the infimum over all limited algorithms of the competitive ratio of that algorithm.

If the competitive ratio of a problem is $O(1)$ then this is generally considered a positive result. Resource augmentation is an analysis technique for those problems where the competitive ratio is $\omega(1)$. In resource augmentation analysis, the limited algorithm is given more resources, in a scheduling problem this might be faster processors or more processors, than the adversary.

I will explain the rationale behind resource augmentation analysis, and give a feel for the types of results that one can obtain with this analysis. I will concentrate on scheduling problems. The problems that I will consider will be something like:

- The classic paging/caching problem. As far as I know, the analysis of this problem by Sleator and Tarjan in [10] was the first use of resource augmentation analysis.
- The online version of the real-time scheduling problem $1|r_i, pmtn|\sum w_i(1-U_i)$ [5,6].
- The online version of the classic uni-processor CPU scheduling problem $1|r_i, pmtn|\sum F_i$ [5,9,2,3]. The analysis of this problem, and the previous problem, in [5] was the first application of resource augmentation to scheduling problems.
- Hard real-time scheduling of preemptable tasks on a multi-processor [9,7]. In [9] the term resource augmentation was coined.
- Jitter control in QoS networks [8].
- The offline version of generalized paging (this is the type of paging that your Netscape or Internet Explorer browser has to perform) [1].

[1] S. Albers, S. Arora, and S. Khanna, "Page replacement for generalized caching problems", ACM/SIAM Symposium on Discrete Algorithms, 31 – 40, 1999.

[2] P. Berman and C. Coulston, "Speed is more powerful than clairvoyance", Scandinavian Workshop on Algorithms and Theory, 255 -263, 1998.

[3] J. Edmonds, "Scheduling in the dark", to appear in ACM Symposium on Theory of Computing.

[4] B. Kalyanasundaram, and K. Pruhs, "The online transportation problem", *European Symposium on Algorithms*, 484–493, 1995.

[5] B. Kalyanasundaram, and K. Pruhs, "Speed is more powerful than clairvoyance", *IEEE Symposium on Foundations of Computer Science*, 1995.

[6] B. Kalyanasundaram, and K. Pruhs, "Maximizing job completions online", European Symposium on Algorithms, 235 – 246, 1998.

[7] T. Lam and K. To, "Trade-offs between speed and processor in hard-deadline scheduling", ACM/SIAM Symposium on Discrete Algorithms, 623 – 632.

[8] Y. Mansour, and B. Patt-Shamir, "Jitter control in QoS networks" IEEE Symposium on Foundations of Computer Science, 50 – 59, 1998.

[9] C. Phillips, C. Stein, E. Torng, and J. Wein, "Optimal time-critical scheduling via resource augmentation", ACM Symposium on Theory of Computing, 140–149, 1997.

[10] D. Sleator, and R. Tarjan, "Amortized efficiency of list update and paging rules", *Communications of the ACM*, **28**, 202 – 208, 1985.

**Simple competitive request scheduling strategies**
MARCO RIEDEL

This work studies the problem of scheduling real-time requests in distributed data servers. We assume the time to be divided into time steps of equal length called rounds. During every round a set of requests arrives at the system, and every resource is able to fulfill one request per round. Every request specifies two distinct resources and requires to get access to one of them. Furthermore, every request has a deadline of $d$, i.e. a request that arrives in round $t$ has to be fulfilled during round $t + d - 1$ at the latest. The goal is to maximize the number of requests that are fulfilled before their deadlines expire.

We examine the scheduling problem in an online setting, i.e. new requests continuously arrive at the system, and an scheduling algorithm has to determine online an assignment of the requests to the resources in such a way that every resource has to fulfill at most one request per round. The output of such an online algorithm in round $t$ is the set of requests which are scheduled to be served in round $t$. We study both global (i.e. centralized) and local (i.e. distributed) scheduling strategies. Competitive analyses is applied in order to measure the performance. Previously, no non-trivial bounds have been known for the

competitive ratio of scheduling strategies in our model. We present (partly matching) upper and lower bounds for several simple scheduling strategies.

Joint work with Petra Berenbrink and Christian Scheideler.

**Approximating the throughput of multiple machines in real-time scheduling**
BARUCH SCHIEBER

We consider the following fundamental scheduling problem. The input to the problem consists of $n$ jobs and $k$ machines. Each of the jobs is associated with a release time, a deadline, a weight, and a processing time on each of the machines. The goal is to find a schedule that maximizes the weight of jobs that meet their respective deadlines. We give constant factor approximation algorithms for four variants of the problem, depending on the type of the machines (identical vs. unrelated), and the weight of the jobs (identical vs. arbitrary). All these variants are known to be NP-Hard, and the two variants involving unrelated machines are also MAX-SNP hard. The specific results obtained are:

- For identical job weights and unrelated machines: a greedy 2-approximation algorithm.
- For identical job weights and $k$ identical machines: the same greedy algorithm achieves a tight $\frac{(1+1/k)^k}{(1+1/k)^k-1}$-approximation factor.
- For arbitrary job weights and a single machine: an LP formulation achieves a 2-approximation for polynomially bounded integral input and a 3-approximation for arbitrary input. For unrelated machines, the factors are 3 and 4 respectively.
- For arbitrary job weights and $k$ identical machines: the LP based algorithm applied repeatedly achieves a $\frac{(1+1/k)^k}{(1+1/k)^k-1}$ approximation factor for polynomially bounded integral input and a $\frac{(1+1/2k)^k}{(1+1/2k)^k-1}$ approximation factor for arbitrary input.
- For arbitrary job weights and unrelated machines: a combinatorial $(3+2\sqrt{2} \approx 5.828)$-approximation algorithm.

Open problems: For identical job weights and a single machine can the factor 2 approximation be improved. Can this problem be shown to be Max-SNP Hard?

Joint work with Amotz Bar-Noy, Sudipto Guha, and Joseph (Seffi) Naor.

**On-line scheduling on parallel machines with a makespan related criterion**
UWE SCHWIEGELSHOHN

In this talk we consider a system consisting of $m$ identical machines on which $n$ jobs must be scheduled. These jobs may be either sequential or parallel requiring a fixed number machines during their whole execution. Further, the jobs may either be run

to completion or preemption (gang scheduling) is allowed. Therefore, we address four different cases. There are two on-line aspects in the problem: The jobs are unknown before their submission and the execution time of a job becomes only known once the execution of the job has completed. We compare the makespan results with the results of so called ratio scheduling. Ratio scheduling is a version of weighted completion time scheduling where the weight of each job is the product of its execution time and its degree of parallelism.

Contrary to makespan scheduling every job contributes to the scheduling costs in ratio scheduling. In both methods every single processor schedule without holes is optimal. However, an optimal makespan schedule need not be an optimal ratio schedule and vice versa. But while an optimal ratio schedule has a close to optimal makespan there are some optimal makespan schedules with high ratio costs in the presence of parallel jobs. These results are best described by the performance guarantees for the various cases:

| | Run to completion | | Preemption | |
|---|---|---|---|---|
| | Makespan | Ratio | Makespan | Ratio |
| Sequential jobs | $2 - \frac{1}{m}$<br>Garey, Graham | 1.5<br>Kawaguchi, Kyan | $2 - \frac{1}{m}$<br>McNaughton | 1.5 |
| Parallel jobs | $2 - \frac{1}{m}$<br>Garey, Graham | $O(m^{0.5})$ | $2 - \frac{1}{m}$<br>McNaughton | 3.56 |

**Page replication — variations on a theme**
RUDOLF FLEISCHER, WŁODZIMIERZ GŁAZEK, AND STEVE SEIDEN

In the online page replication problem, one must decide which nodes of a given network should have a copy of a fixed database (or page). Initially only one node has this page. The page can be replicated to any other node, but only at high cost. Users of the network make requests for data at their respective nodes. How do we best serve these requests? If the page is located close by, the requests can be easily served, but if not we may want to copy the page to a location close to where we are currently receiving requests.

This problem was introduced by Black and Sleator [2], and further studied by Albers and Koga [1], and Głazek [3]. Formally, it can be stated as follows: We have a fixed finite metric $M$ with a distinguished origin vertex $s$. A sequence of requests is given. Before serving each request, the online algorithm has the option of duplicating the page. I.e., one can copy the database from any current vertex $x$ which has it, to another vertex $y$ which does not at cost $D \cdot d(x, y)$ where $D$ is the replication factor. Then the request is served at cost $d(r, x)$ where $r$ is the request vertex and $x$ is the closest vertex which has the page. Most research on this problem has focused on two types of metric spaces: trees and rings.

We present several new results about this problem:

- In their seminal paper about the page replication problem, Black and Sleator [2] claim that 2.5 is a lower bound for "the four node cycle", but give no proof. We give evidence against their claim, showing an upper bound of 2.36603 for the ring with four evenly spaced nodes.
- We show that 2.36603 is also a deterministic lower bound for the 4-node cycle.
- We present the first randomized lower bound for rings; no algorithm is better than 1.75037-competitive for the 4-node cycle.
- We initiate the study of replication in continuous metric spaces. Deterministic algorithms in continuous metric spaces correspond naturally to randomized algorithms in discrete metric spaces with the same competitive ratio. We give a 1.58198-competitive deterministic algorithm for continuous trees and a 2.54150-competitive deterministic algorithm for continuous rings. These algorithms correspond to the randomized discrete algorithms by Albers and Koga [1] and Głazek [3], respectively, but our proofs are much simpler.
- We show a deterministic lower bound of 2.31023 for the continuous cycle.
- We investigate a randomized algorithm for the ring proposed by Albers and Koga [1]. They showed that this algorithm is 3.16396-competitive. We present a modification of their algorithm which is 2.37297-competitive. This is the best known upper bound for the ring.

[1] ALBERS, S., AND KOGA, H. New on-line algorithms for the page replication problem. *Journal of Algorithms 27*, 1 (Apr 1998), 75–96.

[2] BLACK, D. L., AND SLEATOR, D. D. Competitive algorithms for replication and migration problems. Tech. Rep. CMU-CS-89-201, Department of Computer Science, Carnegie-Mellon University, 1989.

[3] GŁAZEK, W. On-line algorithms for page replication in rings. Presented at the *ALCOM Workshop on On-line Algorithms*, Udine, Italy, Sep 1998. Final version to appear in *Theoretical Computer Science*.

**Approximation schemes for scheduling on uniformly related and identical parallel machines**

JIŘÍ SGALL

We give a polynomial approximation scheme for the problem of scheduling on uniformly related parallel machines for a large class of objective functions that depend only on the

machine completion times, including minimizing the $l_p$ norm of the vector of completion times. This generalizes and simplifies many previous results in this area.

Joint work with Leah Epstein.

### Running a job on a collection of dynamic machines, with on-line restarts

ROB VAN STEE

We consider the problem of running a background job on a selected machine of a collection of machines. Each of these machines may become temporarily unavailable (busy) without warning, in which case the scheduler is allowed to restart the job on a different machine. The behaviour of machines is characterized by a Markov chain. The objective is to minimize completion time of the job. For several types of Markov chains, we present elegant and optimal policies.

Joint work with Han La Poutré.

### Competitive advertising

ANDREW TOMKINS

We analyze algorithms for two problems that arise in the context of the World Wide Web. The first, called *recommendation systems*, is exemplified by the problem of suggesting books to users browsing at amazon.com. A recommendation system tracks past actions of a group of users to make recommendations to individual members of the group. The growth of computer-mediated marketing and commerce has led to increased interest in such systems. We introduce a simple analytical framework for recommendation systems, including a basis for defining the utility of such a system. We perform probabilistic analyses of algorithmic methods within this framework. These analyses yield insights into how much utility can be derived from the memory of past actions and on how this memory can be exploited. (A paper describing this work appeared in FOCS 98.)

The second problem we consider, *targeting Markov segments*, occurs when an organization wishes to make use of information about the trails followed by users through a set of states (for example, pages on the web). Consider two user populations, of which one is *targeted* and the other is not. Users in the targeted population follow a Markov chain $P$ on a space of $n$ states. The untargeted population follows $Q$, also defined on the same set of $n$ states. Each time a user arrives at a state, she is presented an advertisement with some probability. Presenting the ad incurs a cost proportional to the total flow (targeted and untargeted) through the state, and generates revenue proportional to the flow of targeted users through the state.

The world-wide web is a natural setting for such a problem. Internet Service Providers (ISP's) have trail information for building such Markovian user models. In this paper we study the simple problem above, as well as the variants with multiple targetable segments. In some settings the policy need not be a *static* probability distribution on

states. Instead, we can dynamically vary the policy based on the user's path through the states. We provide characterizations which reveal interesting insight into the nature of optimal policies, and then use these insights for algorithm design. Such targeting problems do not seem amenable to solutions using methods from familiar fields such as Markov decision processes. (A paper describing this work appeared in STOC 99.)

This is joint work with Moses Charikar, Ravi Kumar, Prabhakar Raghavan, and Sridhar Rajagopalan.

**Mathematical programming and lower bounds for online problems**
Eric Torng

In this paper, we present an algorithm for generating lower bounds for many online problems using mathematical programming techniques. This result is fairly unique as lower bounds are typically developed in an ad hoc fashion. Our algorithm can be applied to a wide variety of problems including several variations of the $k$-server problem, the online load balancing problem, and the online circuit routing problem. One of our results is that the complement of the $k$-server conjecture is recursively enumerable. That is, if the $k$-server conjecture is not true, our algorithm is guaranteed to eventually find a proof that the $k$-server conjecture is not true. Our algorithm can also be used to develop essentially optimal online algorithms for many online problems. Its main drawback is its prohibitive running time. Thus we have only derived a slightly improved lower bound of 1.85358 for the online load balancing problem on $m \geq 80$ identical machines.

This is joint work with Todd Gormley.

# Open Problems

**Two routing problems**
YOSSI AZAR

(1) Routing with bandwidth 1/2 (or $1/k$). Given a general network G=(V,E); each edge has unit capacity; requests from $s_i$ to $t_i$ with bandwidth 1/2 (or $1/k$) appear online. Each request should be either rejected or routed on a virtual path. The goal is to maximize the number of accepted requests (subject to capacity constraint). Is their a poly-log competitive randomized algorithm?

Known results: For bandwidth = 1 [6] showed an $n^c$ lower bound. For bandwidth $< 1/\log n$ [3] showed a $\log n$ tight competitive algorithm. For fixed paths [2] showed an $n^{1/(k+1)}$ lower bound. For trees/meshes, [4,5] showed a $\log n$ tight competitive algorithm.

(2) The same problem where accepted request may be discarded later (rejected are lost). Is their a constant competitive algorithm or a non-constant lower bound for low bandwidth requests? Is their a good competitive randomized algorithm or non-constant randomized lower bound for bandwidth $1/k$?

Known results: For bandwidth = 1 [6] showed an $n^c$ lower bound. For bandwidth $< 1/\log n$ the result of [3] of course holds. For fixed paths [2] showed an $n^{1/2(k+1)}$ lower bound. For the line [1] showed a constant competitive algorithm.

[1] R. Adler and Y. Azar, *Beating the logarithmic lower bound: randomized preemptive disjoint paths and call control algorithms*, Proc. 10th ACM-SIAM Symp. on Discrete Algorithms, 1999, pp. 1–10.

[2] N. Alon, U. Arad, and Y. Azar, *Independent sets in hypergraphs with applications to routing via fixed paths*, Proc. 2nd Workshop on Approximation Algorithms for Combinatorial Optimization Problems, 1999, to appear.

[3] B. Awerbuch, Y. Azar, and S. Plotkin, *Throughput-competitive online routing*, 34th IEEE Symposium on Foundations of Computer Science, 1993, pp. 32–40.

[4] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosén, *Competitive non-preemptive call control*, Proc. of 5th ACM-SIAM Symposium on Discrete Algorithms, 1994, pp. 312–320.

[5] B. Awerbuch, R. Gawlick, T. Leighton, and Y. Rabani, *On-line admission control and circuit routing for high performance computation and communication*, Proc. 35th IEEE Symp. on Found. of Comp. Science, 1994, pp. 412–423.

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

**The CNN problem**
ELIAS KOUTSOUPIAS

Consider a server in the Euclidean plane that responds to a sequence of requests as in the $k$-server problem. However, for this problem, the server does not have to move to the requested point; it can service a request by moving to any point that is on the same horizontal or vertical line with the request. The name —suggested by Gerhard Woeginger— is derived from a certain television network. The relation is that a TV crew can cover incidents in Manhattan (by zooming in) from far away, provided that it is in the same street; in the worst case, incidents happen only at intersections!

The problem was originally suggested by Mike Zaks and studied by William Burley. The best lower bound of the competitive ratio is $6 + \sqrt{17} = 10.123\ldots$ No upper bound is known. I conjecture that the generalized WFA $(3w(A') + d(A, A'))$ has competitive ratio around 13. Special cases and certain generalizations of the problem seem to be important on-line problems. For example, the "cow-path problem" and the "weighted 2-server problem in a line" are special cases of the CNN problem.

In the near future, I plan to put more information about this problem at the webpage `http://www.cs.ucla.edu/~elias/publications`

**Using a buffer as a Cache in the (read only) file migration problem**
KETSIYA E. MEIRMAN

In the file migration problem [1,2,3,4] for a uniform graph, there are $n$ connected nodes that access a common file, that consists of $D$ pages. Assume that there is only one copy of the file, and that it can be migrated from one node to another. A sequence of requests is generated at the different nodes accesses to the file. In the classical file migration problem, each node maintains a read/write buffer of one page. If the file is present in a node, then the local access cost of the file is 0. When a process access a file from a remote node, then this request can be performed in two ways: either by bringing one page from the file to the buffer of the requesting node, at the cost of 1, or by migrating the whole file to the requesting node, at cost D. In particular, we note that in the first case, each remote file access results in an actual read operation from the remote node, even if the requested page is already in the buffer from the previous operation.

An interesting variation of the file migration problem is for read-only files. In this

case if the requested page is already in the buffer of a node, then there is no need to read that page again from the file, i.e., the buffer can be considered as a cache, which is valid as long as successive read requests are addressed to the same page. This problem can further be generalized to nodes with buffer size $k$, where $1 \leq k < D$.

This scenario, of accessing successively many time the same page is realistic, since there is often locality of reference in file accesses, both since one item of a page is accessed more then once or since multiple items of one page are accessed successively.

For example, assume that there are two nodes: $P_1$ and $P_2$, and a read-only file, with D pages $f_1, ..., f_D$ that reside in either $P_1$ or $P_2$. A request $\sigma_{P_i,f_j}$ denotes that there is a request access from processor $P_i$ to the $j$'th page of the file. For the access sequence:

$$\sigma_{P_1,f_1}, \sigma_{P_1,f_1}, \sigma_{P_2,f_1}, \sigma_{P_1,f_1}, \sigma_{P_1,f_1}, \sigma_{P_2,f_2}, ..., \sigma_{P_1,f_1}, \sigma_{P_1,f_1}, \sigma_{P_2,f_{i \bmod D}}$$

it is better to place the file in $P_2$ , since the process in $P_1$ can always use the cache page $f_1$ that is brought only once.

[1] Bartal Y., Disributed Paging. In Online Algorithms, Fiat, A.; Woeginger G.J.; Lecture Notes in Computer Science, Vol 1442, Springer, May 1998.

[2] Bartal, Y.; Charikar, M.; Indyk P., On Page Migration and Othe Relaxed Task Systems. In Proc. of the 8th Ann. ACM-SIAM Symp. on Discrete Algorithms, pages 43-52, January 1997.

[3] Black, D.L. and Sleator D.D., Competitive Algorithms for Replication and Migration Problems. In Technical report CMU-CS-89-201, Department of Computer Science, Carnegie Mellon University, November 1989.

[4] Chrobak, M.; Larmore, L.L.; Reingold, N. and Westbrook J., Page Migration Algorithms Using Work Functions. In Proceedings of the 4th International Symposium on Algorithms and Computation, ISAAC '93, volume 762 of Lecture Notes in Computer Science, pages 406-415, Hong Kong, Springer-Verlag, 1993.

## An online sorting problem

BALA KALYANASUNDARAM AND KIRK PRUHS

We ran into the following basic online combinatorial problem doing research on online dynamic storage allocation that we imagine could well arise in other applications and that seems interesting.

The online algorithm is given an online sequence $S = s_1, \ldots, s_i$ of $n$ integers. The online algorithm must irrevocably place $s_i$ in an array $A$ of size $m \geq n$ before seeing $s_{i+1}$. The goal is that $A$ should be as "sorted" as possible. One can have many variants of this problem. There are many possible formalizations of "sorted". In the particular problem that we were examining, we wanted that the largest $s_i$ should near each other in $A$. But there are probably many more natural definitions of "sorted". The online algorithm may

be deterministic or randomized. The input $S$ may be adversarial, or each $s_i$ could be selected from some independent probability distribution. In seems that in many variants of this problem the online algorithm has a very difficult task. So top level, perhaps the goal should be to determine in what variants of the problem can the online algorithm can do something better than random assignment.

### Algorithms for on-line scheduling on related machines

Jiří Sgall

Consider the following model of scheduling. We have $m$ machines of possibly different speeds and a sequence of $n$ jobs arriving on-line one by one. As soon as a job arrives, we learn its size (processing time) and we have to assign it to one or more machines and time slots; we are allowed to use preemption, i.e., to use more time slots on one or different machines, provided that the time intervals are non-overlapping and sufficient to process the jobs. The goal is to minimize the makespan (the total length of the schedule).

We have a lower bound approaching 2 as the number of machines increases to infinity. We also have tight bounds for 2 machines and any speeds.

We conjecture that the optimal competitive ratio is fairly small, probably at most 4. However, at this point we do not have any better algorithms than the non-preemptive ones, which (if compared to the optimal preemptive schedule) yields only a competitive ratio around 20. So, the open problem is to find any better algorithms.

For more details see `http://www.math.cas.cz/~sgall/ps/rel-lb.ps`