JUNE 1997

OPTIMA

Mathematical Programming Society Newsletter

No 54

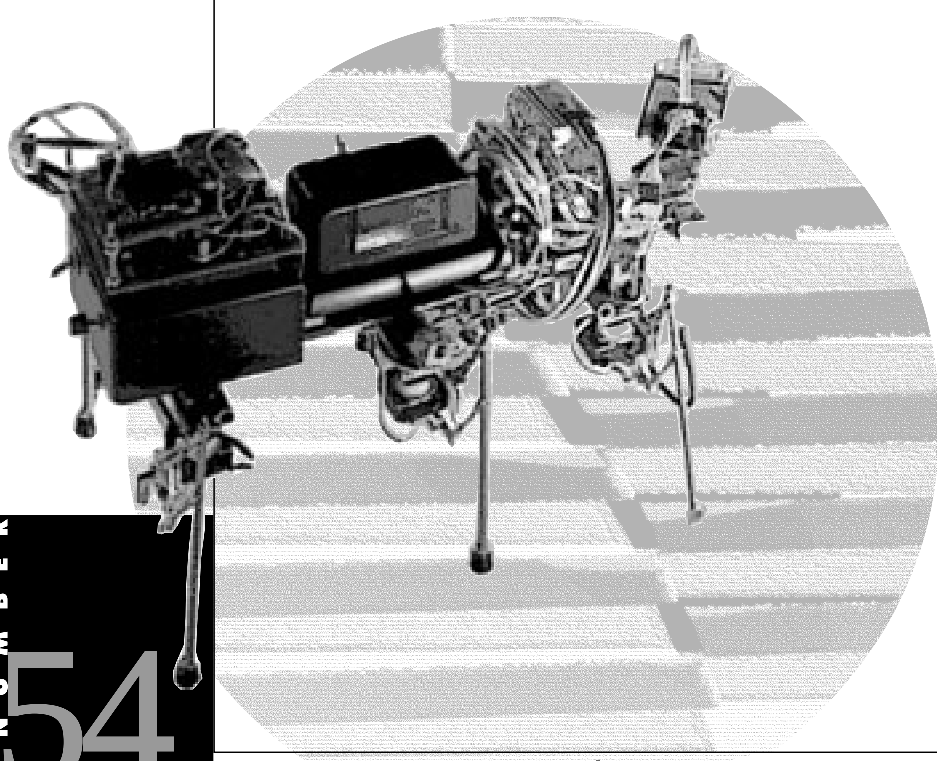
Competitive Online Algorithms

Susanne Albers
Max-Planck-Institut für Informatik,
Im Stadtwald, 66123 Saarbrücken,
Germany
E-mail: albers@mpi-sb.mpg.de

OVERVIEW

Over the past ten years, online algorithms have received considerable research interest. Online problems had been investigated already in the seventies and early eighties but an extensive, systematic study only started when Sleator and Tarjan (1985) suggested comparing an online algorithm to an optimal offline algorithm and Karlin, Manasse, Rudolph and Sleator (1988) coined the term *competitive analysis*. In this article we give an introduction to the theory of online algorithms and survey interesting application areas. We present important results and outline directions for future research.

SEE PAGE TWO ▶



NUMBER 54

Introduction

The traditional design and analysis of algorithms assumes that an algorithm, which generates an output, has complete knowledge of the entire input. However, this assumption is often unrealistic in practical applications. Many of the algorithmic problems that arise in practice are *online*. In these problems the input is only partially available because some relevant input data will arrive in the future and is not accessible at present. An online algorithm must generate an output without knowledge of the entire input. Online problems arise in areas such as resource allocation in operating systems, data-structuring, distributed computing, scheduling, and robotics. We give some illustrative examples.

PAGING: In a two-level memory system, consisting of a small fast memory and a large slow memory, a paging algorithm has to keep actively referenced pages in fast memory without knowing which pages will be requested in the future.

DISTRIBUTED DATA MANAGEMENT: A set of files has to be distributed in a network of processors, each of which has its own local memory. The goal is to dynamically re-allocate files in the system so that a sequence of read and write requests can be processed with low communication cost. It is unknown which files a processor will access in the future.

MULTIPROCESSOR SCHEDULING: A sequence of jobs must be scheduled on a given set of machines. Jobs arrive one by one and must be scheduled immediately without knowledge of future jobs.

NAVIGATION PROBLEMS IN ROBOTICS: A robot is placed in an unknown environment and has to find a short path from a point s to a point t. The robot learns about the environment as it travels through the scene.

We will address these problems in more detail in the following sections.

In recent years, it has been shown that *competitive analysis* is a powerful tool to analyze the performance of online algorithms. The idea of competitiveness is to compare the output generated by an online algorithm to the output produced by an *offline* algorithm. An offline algorithm is an omniscient algorithm that knows the entire input data and can compute an optimal output. The better an online algorithm approximates the optimal solution, the more competitive this algorithm is.

Basic concepts

Formally, many online problems can be described as follows. An online algorithm A is presented with a *request sequence* $\sigma = \sigma(1), \sigma(2), \ldots, \sigma(m)$. The requests $\sigma(t)$, $1 \le t \le m$, must be served in their order of occurrence. More specifically, when serving request $\sigma(t)$, algorithm A does not know any request $\sigma(t')$ with $t' > t$. Serving requests incurs cost, and the goal is to minimize the total cost paid on the entire request sequence. This setting can also be regarded as a *request-answer* game: An adversary generates requests, and an online algorithm has to serve them one at a time.

To illustrate this formal model we reconsider the paging problem, which is one of the most fundamental online problems and start with a precise definition.

THE PAGING PROBLEM: Consider a two-level memory system that consists of a small fast memory and a large slow memory. Each request specifies a page in the memory system. A request is served if the corresponding page is in fast memory. If a requested page is not in fast memory, a *page fault* occurs. Then a page must be moved from fast memory to slow memory so that the requested page can be loaded into the vacated location. A paging algorithm specifies which page to evict on a fault. If the algorithm is online, then the decision of which page to evict must be made without knowledge of any future requests. The cost to be minimized is the total number of page faults incurred on the request sequence.

Sleator and Tarjan [64] suggested evaluating the performance on an online algorithm using *competitive analysis*. In a competitive analysis, an online algorithm A is compared to an *optimal offline algorithm*. An optimal offline algorithm knows the entire request sequence in advance and can serve it with minimum cost. Given a request sequence σ, let $C_A(\sigma)$ denote the cost incurred by A and let $C_{OPT}(\sigma)$ denote the cost incurred by an optimal offline algorithm OPT. The algorithm A is called c-competitive if there exists a constant a such that

$$C_A(\sigma) \le c \cdot C_{OPT}(\sigma) + a$$

for all request sequences σ. Here we assume that A is a deterministic online algorithm. The factor c is also called the *competitive ratio* of A.

With respect to the paging problem, there are three well-known deterministic online algorithms.

LRU (Least Recently Used): On a fault, evict the page in fast memory that was requested least recently.

FIFO (First-In First-Out): Evict the page that has been in fast memory longest.

LFU (Least Frequently Used): Evict the page that has been requested least frequently.

Let k be the number of memory pages that can simultaneously reside in fast memory. Sleator and Tarjan [64] showed that the algorithms LRU and FIFO are k-competitive. Thus, for any sequence of requests, these algorithms incur at most k times the optimum number of page faults. Sleator and Tarjan also proved that no deterministic online paging algorithm can achieve a competitive ratio smaller than k. Hence, both LRU and FIFO achieve the best possible competitive ratio. It it easy to prove that LFU is not competitive for any constant c.

An optimal offline algorithm for the paging problem was presented by Belady [19]. The algorithm is called MIN and works as follows.

MIN: On a fault, evict the page whose next request occurs furthest in the future.

Belady showed that on any sequence of requests, MIN achieves the minimum number of page faults.

It is worth noting that the competitive ratios shown for deterministic paging algorithms are not very meaningful from a practical point of view. The performance ratios of LRU and FIFO become worse as the size of the fast memory increases. However, in practice, these algorithms perform better the larger the fast memory is. Furthermore, the competitive ratios of LRU and FIFO are the same, whereas in practice LRU performs much better. For these reasons, there has been a study of competitive paging algorithms with locality of reference. We discuss this issue in the last section.

A natural question is: Can an online algorithm achieve a better competitive ratio if it is allowed to use randomization?

The competitive ratio of a randomized online algorithm *A* is defined with respect to an adversary. The adversary generates a request sequence σ and it also has to serve σ. When constructing σ, the adversary always knows the description of *A*. The crucial question is: When generating requests, is the adversary allowed to see the outcome of the random choices made by *A* on previous requests?

Ben-David *et al*. [20] introduced three kinds of adversaries.

Oblivious Adversary: The oblivious adversary has to generate a complete request sequence in advance, before any requests are served by the online algorithm. The adversary is charged the cost of the optimum offline algorithm for that sequence.

Adaptive Online Adversary: This adversary may observe the online algorithm and generate the next request based on the algorithm's (randomized) answers to all previous requests. The adversary must serve each request online, i.e., without knowing the random choices made by the online algorithm on the present or any future request.

Adaptive Offline Adversary: This adversary also generates a request sequence adaptively. However, it is charged the optimum offline cost for that sequence.

A randomized online algorithm *A* is called *c*-competitive against any oblivious adversary if there is a constant *a* such for all request sequences σ generated by an oblivious adversary, $E[C_A(\sigma)] \le c \cdot C_{OPT}(\sigma) + a$. The expectation is taken over the random choices made by *A*.

Given a randomized online algorithm *A* and an adaptive online (adaptive offline) adversary ADV, let $E[C_A]$ and $E[C_{ADV}]$ denote the expected costs incurred by *A* and ADV in serving a request sequence generated by ADV. A randomized online algorithm *A* is called *c*-competitive against any adaptive online (adaptive off-line) adversary if there is a constant *a* such that for all adaptive online (adaptive offline) adversaries ADV, $E[C_A] \le c \cdot E[C_{ADV}] + a$, where the expectation is taken over the random choices made by *A*.

Ben-David *et al*. [20] investigated the relative strength of the adversaries and showed the following statements.

1. If there is a randomized online algorithm that is *c*-competitive against any adaptive offline adversary, then there also exists a *c*-competitive deterministic online algorithm.

2. If *A* is a *c*-competitive randomized algorithm against any adaptive online adversary and if there is a *d*-competitive algorithm against any oblivious adversary, then *A* is $(c \cdot d)$-competitive against any adaptive offline adversary.

Statement 1 implies that randomization does not help against the adaptive offline adversary. An immediate consequence of the two statements above is:

3. If there exists a *c*-competitive randomized algorithm against any adaptive online adversary, then there is a c^2-competitive deterministic algorithm.

Against oblivious adversaries, randomized online paging algorithms can considerably improve the ratio of *k* shown for deterministic paging. The following algorithm was proposed by Fiat *et al*. [39].

Marking: The algorithm processes a request sequence in phases. At the beginning of each phase, all pages in the memory system are unmarked. Whenever a page is requested, it is *marked*. On a fault, a page is chosen uniformly at random from among the unmarked pages in fast memory, and that page is evicted. A phase ends when all pages in fast memory are marked and a page fault occurs. Then, all marks are erased and a new phase is started.

Fiat *et al*. [39] analyzed the performance of the Marking algorithm and showed that it is $2H_k$-competitive against any oblivious adversary, where $H_k = \sum_{i=1}^{k} 1/i$ is the *k*-th Harmonic number. Note that H_k is roughly ln *k*.

Fiat *et al*. [39] also proved that no randomized online paging algorithm against any oblivious adversary can be better than H_k-competitive. Thus the Marking algorithm is optimal, up to a constant factor. More complicated paging algorithms achieving an optimal competitive ratio of H_k were given in [57,1].

Self-organizing data structures

The *list update problem* is one of the first online problems that were studied with respect to competitiveness. The problem is to maintain a set of items as an unsorted linear list. We are given a linear linked list of items. As input we receive a request sequence σ, where each request specifies one of the items in the list. To serve a request a list update algorithm must *access* the requested item, i.e., it has to start at the front of the list and search linearly through the items until the desired item is found. Serving a request to the item that is stored at position *i* in the list incurs a cost of *i*. While processing a request sequence, a list update algorithm may rearrange the list. Immediately after an access, the requested item may be moved at no extra cost to any position closer to the front of the list. These exchanges are called *free exchanges*. Using free exchanges, the algorithm can lower the cost on subsequent requests. At any time two adjacent items in the list may be exchanged at a cost of 1. These exchanges are called *paid exchanges*.

With respect to the list update problem, we require that a *c*-competitive online algorithm has a performance ratio of *c for all size lists*. More precisely, a deterministic online algorithm for list update is called *c*-competitive if there is a constant *a* such that for all size lists and all request sequences σ, $C_A(\sigma) \le c \cdot C_{OPT}(\sigma) + a$.

Linear lists are one possibility of representing a set of items. Certainly, there are other data structures such as balanced search trees or hash tables that, depending on the given application, can maintain a set in a more efficient way. In general, linear lists are useful when the set is small and consists of only a few dozen items. Recently, list update techniques have been applied very successfully in the development of data compression algorithms [21,28].

There are three well-known deterministic online algorithms for the list update problem.

Move-To-Front: Move the requested item to the front of the list.

Transpose: Exchange the requested item with the immediately preceding item in the list.

Frequency-Count: Maintain a frequency count for each item in the list. Whenever an item is requested, increase its count by 1. Maintain the list so that the items always occur in nonincreasing order of frequency count.

Sleator and Tarjan [64] proved that Move-To-Front is 2-competitive. Karp and Raghavan [48] observed that no deterministic online algorithm for list update can have a competitive ratio smaller than 2. This implies that Move-To-Front achieves the best possible competitive ratio. Sleator and Tarjan also showed that Transpose and Frequency-Count are not *c*-competitive for any constant *c* independent of the list length. Thus, in terms of competitiveness, Move-To-Front is superior to Transpose and Frequency-Count.

Next we address the problem of randomization in the list update problem. Against adaptive adversaries, no randomized online algorithm for list update can be better than 2-competitive, see [20,62]. Thus we concentrate on algorithms against oblivious adversaries.

We present the two most important algorithms. Reingold *et al*. [62] gave a very simple algorithm, called Bit.

Bit: Each item in the list maintains a bit that is complemented whenever the item is accessed. If an access causes a bit to change to 1, then the requested item is moved to the front of the list. Otherwise the list remains unchanged. The bits of the items are initialized independently and uniformly at random.

Reingold it *et al*. [62] proved that Bit is 1.75-competitive against oblivious adversaries. The best randomized algorithm currently known is a combination of the Bit algorithm and a deterministic 2-competitive online algorithm called Timestamp proposed in [2].

Timestamp (TS): Insert the requested item, say *x*, in front of the first item in the list that precedes *x* and that has been requested at most once since the last request to *x*. If there is no such item or if *x* has not been requested so far, then leave the position of *x* unchanged.

As an example, consider a list of six items being in the order $L : x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow x_5 \rightarrow x_6$. Suppose that algorithm TS has to serve the second request to x_5 in the request sequence $\sigma = \ldots x_5, x_2, x_2, x_3, x_1, x_1, x_5$. Items x_3 and x_4 were requested at most once since the last request to x_5, whereas x_1 and x_2 were both requested twice. Thus, TS will insert x_5 immediately in front of x_3 in the list.

A combination of Bit and TS was proposed by [5].

Combination: With probability 4/5 the algorithm serves a request sequence using Bit, and with probability 1/5 it serves a request sequence using TS.

This algorithm is 1.6-competitive against oblivious adversaries [5]. The best lower bound currently known is due to Teia [67]. He showed that if a randomized list update algorithm is *c*-competitive against oblivious adversaries, then $c \geq 1.5$.

An interesting open problem is to give tight bounds on the competitive ratio that can be achieved by randomized online algorithms against oblivious adversaries.

Many of the concepts shown for self-organizing linear lists can be extended to binary search trees. The most popular version of self-organizing binary search trees are the *splay trees* presented by Sleator and Tarjan [65]. In a splay tree, after each access to an element *x* in the tree, the node storing *x* is moved to the root of the tree using a special sequence of rotations that depends on the structure of the access path. This reorganization of the tree is called *splaying*.

Sleator and Tarjan [65] analyzed splay trees and proved a series of interesting results. They showed that the amortized asymptotic time of access and update operations is as good as the corresponding time of balanced trees. More formally, in an *n*-node splay tree, the amortized time of each operation is $O(\log n)$. It was also shown [65] that on any sequence of accesses, a splay tree is as efficient as the optimum static search tree. Moreover, Sleator and Tarjan [65] presented a series of conjectures, some of which have been resolved or partially resolved [31,32,33,66]. On the other hand, the famous splay tree conjecture is still open: It is conjectured that on any sequence of accesses splay trees are as efficient as any dynamic binary search tree.

The k-server problem

The *k*-server problem is one of the most fundamental and extensively studied online problems. In the *k*-server problem we are given a metric space *S* and *k* mobile servers that reside on points in *S*. Each request specifies a point $x \in S$. To serve a request, one of the *k* servers must be moved to the requested point unless a server is already present. Moving a server from point *x* to point *y* incurs a cost equal to the distance between *x* and *y*. The goal is to serve a sequence of requests so that the total distance traveled by all servers is as small as possible.

The *k*-server problem contains paging as a special case. Consider a metric space in which the distance between any two points in 1; each point in the metric space represents a page in the memory system and the pages covered by servers are those that reside in fast memory. The *k*-server problem also models more general caching problems, where the cost of loading an item into fast memory depends on the size of the item. Such a situation occurs, for instance, when font files are loaded into the cache of a printer. More generally, the *k*-server problem can also be regarded as a vehicle routing problem.

The *k*-server problem was introduced by Manasse *et al*. [56] in 1988 who also showed a lower bound for deterministic *k*-server algorithms: Let *A* be a deterministic online *k*-server algorithm in an arbitrary metric space. If *A* is *c*-competitive, then $c \geq k$.

Manasse *et al*. also conjectured that there exists a deterministic *k*-competitive online *k*-server algorithm. Only recently, Koutsoupias and Papadimitriou [52] showed that there is a (2*k*-1)-competitive algorithm. Before, *k*-competitive algorithms were known for special metric spaces (e.g. trees [30] and resistive spaces [34] and special values of *k* (*k*=2 and *k* = *n*-1, where *n* is the number of points in the metric space [56]. It is worthwhile to note that the greedy algorithm, which always moves the closest server to the requested point, is not competitive.

The algorithm analyzed by Koutsoupias and Papadimitriou is the Work Function algorithm. Let *X* be a configuration of the servers. Given a request sequence $\sigma = \sigma(1), \ldots, \sigma(t)$, the *work function* $w(X)$ is the minimal cost of serving σ and ending in configuration *X*.

WORK FUNCTION: Suppose that the algorithm has served $\sigma = \sigma(1), \ldots, \sigma(t\text{-}1)$ and that a new request $r = \sigma(t)$ arrives. Let X be the current configuration of the servers and let x_i be the point where server s_i, $1 \leq i \leq k$, is located. Serve the request by moving the server s_i that minimizes $w(X_i) + dist(x_i,r)$, where $X_i = X - \{x_i\} + \{r\}$.

Koutsoupias and Papadimitriou [52] proved that the WORK FUNCTION algorithm is $(2k\text{-}1)$-competitive in an arbitrary metric space. An interesting open problem is to show that the WORK FUNCTION algorithm is indeed k-competitive or to develop another deterministic online k-server algorithm that achieves a competitive ratio of k.

An elegant randomized rule for moving servers was proposed by Raghavan and Snir [61].

HARMONIC: Suppose that there is a new request at point r and that server s_i, $1 \leq i \leq k$, is currently at point x_i. Move server s_i with probability

$$p_i = \frac{1/dist(x_i,r)}{\sum_{j=1}^{k} 1/dist(x_j,r)}$$

to the request.

Intuitively, the closer a server is to the request, the higher the probability that it will be moved. Grove [42] proved that the HARMONIC algorithm has a competitive ratio of $c \leq \frac{5}{4}k \cdot 2^k - 2k$. The competitiveness of HARMONIC is not better than $k(k+1)/2$, see [58]. An open problem is to develop tight bounds on the competitive ratio achieved by HARMONIC.

Recently Bartal *et al.* [14] presented a randomized online algorithm that achieves a competitive ratio of $O(c^6 \log^6 k)$ on metric spaces consisting of $k+c$ points. The main open problem in the area of the k-server problem is to develop randomized online algorithms that have a competitive ratio of $c < k$ in an arbitrary metric space.

Distributed data management

In distributed data management the goal is to dynamically re-allocate memory pages in a network of processors, each of which has its own local memory, so that a sequence of read and write requests to memory pages can be served with low total communication cost. The configuration of the system can be changed by *migrating* and *replicating* a memory page, i.e., a page is moved or copied from one local memory to another.

More formally, page allocation problems can be described as follows. We are given a weighted undirected graph G. Each node in G corresponds to a processor and the edges represent the interconnection network. We generally concentrate on one particular page in the system. We say that *a node v has the page* if the page is contained in v's local memory. *A request at a node v* occurs if v wants to read or write an address from the page. Immediately after a request, the page may be migrated or replicated from a node holding the page to another node in the network. We use the cost model introduced by Bartal *et al.* [18] and Awerbuch *et al.* [8]. (1) If there is a read request at v and v does not have the page, then the incurred cost is $dist(u,v)$, where u is the closest node with the page. (2) The cost of a write request at node v is equal to the cost of communicating from v to all other nodes with a page replica. (3) Migrating or replicating a page from node u to node v incurs a cost of $d \cdot dist(u,v)$, where d is the page size factor. (4) A page replica may be erased at 0 cost. In the following we only consider *centralized* page allocation algorithms, i.e., each node always knows where the closest node holding the page is located in the network.

Bartal *et al.* [18] and Awerbuch *et al.* [8] presented deterministic and randomized online algorithms achieving an optimal competitive ratio of $O(\log n)$, where n is the number of nodes in the graph. We describe the randomized solution [18].

COINFLIP: If there is a read request at node v and v does not have the page, then with probability $\frac{1}{d}$, replicate the page to v. If there is a write request at node v, then with probability $\frac{1}{\sqrt{3d}}$, migrate the page to v and erase all other page replicas.

The *page migration* problem is a restricted problem where we keep only one copy of each page in the entire system. If a page is writable, this avoids the problem of keeping multiple copies of a page consistent. For this problem, constant competitive algorithms are known. More specifically, there are deterministic online migration algorithms that achieve competitive ratios of 7 and 4.1, respectively, see [8,16]. We describe an elegant randomized algorithm due to Westbrook [69].

COUNTER: The algorithm maintains a global counter C that takes integer values in $[0,k]$, for some positive integer k. Counter C is initialized uniformly at random to an integer in $[1,k]$. On each request, C is decremented by 1. If $C = 0$ after the service of the request, then the page is moved to the requesting node and C is reset to k.

Westbrook showed that the COUNTER algorithm is c-competitive, where $c = \max\{2 + \frac{2d}{k}, 1 + \frac{k+1}{2d}\}$. He also determined the best value of k and showed that, as d increases, the best competitive ratio decreases and tends to $1 + \Phi$, where $\Phi = (1+\sqrt{5})/2 \approx 1.62$ is the Golden Ratio.

All of the above solutions assume that the local memories of the processors have infinite capacity. Bartal *et al.* [18] showed that if the local memories have finite capacity, then no online algorithm for page allocation can be better than $\Omega(m)$-competitive, where m is the total number of pages that can be accommodated in the system.

Scheduling and load balancing

The general situation in online *scheduling* is as follows. We are given a set of m machines. A sequence of jobs $\sigma = J_1, J_2, \ldots, J_n$ arrives online. Each job J_k has a processing p_k time that may or may not be known in advance. As each job arrives, it has to be scheduled immediately on one of the m machines. The goal is to optimize a given objective function. There are many problem variants, e.g., we can study various machine types and various objective functions.

We consider one of the most basic settings introduced by Graham [41] in 1966. Suppose that we are given m *identical* machines. As each job arrives, its processing time is known in advance. The goal is to minimize the makespan, i.e., the completion time of the last job that finishes.

Graham [41] proposed the GREEDY algorithm and showed that it is $(2 - \frac{1}{m})$-competitive.

GREEDY: Always assign a new job to the least loaded machine.

In recent years, research has focused on finding algorithms that achieve a competitive ratio c, $c < 2$, for all values of m. In 1992, Bartal *et al.* [17] gave an algorithm that is 1.986-competitive. Karger *et al.* [46] generalized the algorithm and proved an upper bound of 1.945. The best algorithm known so far achieves a competitive ratio of 1.923, see [3].

Next we discuss some extensions of the scheduling problem mentioned above.

IDENTICAL MACHINES, RESTRICTED ASSIGNMENT: We have a set of m identical machines, but each job can only be assigned to one of a subset of admissible machines. Azar *et al.* [12] showed that the GREEDY algorithm, which always assigns a new job to the least loaded machine among the admissible machines, is $O(\log m)$-competitive.

RELATED MACHINES: Each machine i has a speed s_i, $1 \le i \le m$. The processing time of job J_k on machine i is equal to p_k/s_i. Aspnes *et al*. [6] showed that the GREEDY algorithm, that always assigns a new job to a machine so that the load after the assignment is minimized, is $O(\log m)$-competitive. They also presented an algorithm that is 8-competitive.

UNRELATED MACHINES: The processing time of job J_k on machine i is $p_{k,i}$, $1 \le k \le n$, $1 \le i \le m$. Aspnes *et al*. [6] showed that GREEDY is only m-competitive. However, they also gave an algorithm that is $O(\log m)$-competitive.

In online *load balancing* we have again a set of m machines and a sequence of jobs $\sigma = J_1, J_2, \ldots, J_n$ that arrive online. However, each job J_k has a *weight* $w(k)$ and an unknown duration. For any time t, let $l_i(t)$ denote the load of machine i, $1 \le i \le m$, at time t, which is the sum of the weights of the jobs present on machine i at time t. The goal is to minimize the maximum load that occurs during the processing of σ.

We refer the reader to [9] for an excellent survey on online load balancing and briefly mention a few basic results. We concentrate again on settings with m identical machines. Azar and Epstein [9] showed that the GREEDY algorithm is $(2 - \frac{1}{m})$-competitive. The load balancing problem becomes more complicated with *restricted assignments*, i.e., each job can only be assigned to a subset of admissible machines. Azar *et al.* [10] proved that GREEDY achieves a competitive ratio of $m^{2/3}(1+o(1))$. They also proved that no online algorithm can be better than $\Omega(\sqrt{m})$-competitive. In a subsequent paper, Azar *et al*. [11] gave a matching upper bound of $O(\sqrt{m})$.

Robotics

There are three fundamental online problems in the area of robotics.

NAVIGATION: A robot is placed in an unknown environment and has to find a short path from a source point s to a target t.

EXPLORATION: A robot is placed in an unknown environment and has to construct a complete map of that environment using a short path.

LOCALIZATION: The robot has a map of the environment. It "wakes up" at a position s and has to uniquely determine its initial position using a short path.

In the following we concentrate on the robot navigation problem. We refer the reader to [4,35,36,44] for literature on the exploration problem, and to [37,43,51,63] for literature on the localization problem.

Many robot navigation problems were introduced by Baeza-Yates *et al*. [13] and Papadimitriou and Yannakakis [59]. We call a robot navigation strategy A c-competitive, if the length of the path used by A is at most c times the length of the shortest possible path.

First we study a simple setting introduced by Baeza-Yates *et al*. [13]. Assume that the robot is placed on a line. It starts at some point s and has to find a point t on the line that is a distance of n away. The robot is tactile, i.e., it only knows that it has reached the target when it is located on t. Since the robot does not know whether t is located to the left or to the right of s, it should not move a long distance in one direction. After having traveled a certain distance in one direction, the robot should return to s and move in the other direction. For $i = 1,2, \ldots$, let $f(i)$ be the distance walked by the robot before the i-th turn since its last visit to s. Baeza-Yates *et al*. [13] proved that the "doubling" strategy $f(i) = 2^i$ is 9-competitive and that this is the best possible.

A more complex navigation problem is as follows. A robot is placed in a 2-dimensional scene with obstacles. As usual, it starts at some point s and has to find a short path to a target t. When traveling through the scene of obstacles, the robot always knows its current position and the position of t. However, the robot does not know the positions and extents of the obstacles in advance. It learns about the obstacles as it walks through the scene.

Most previous work on this problem has focused on the case that the obstacles are axis-parallel rectangles. Papadimitriou and Yannakakis [59] gave a lower bound. They showed that no deterministic online navigation algorithm in a general scene with n rectangular, axis parallel obstacles can have a competitive ratio smaller than $\Omega(\sqrt{n})$. (In fact, the lower bound also holds for a relaxed problem where the robot only has to reach *some* point on a vertical wall.)

Blum *et al*. [25] developed a deterministic online navigation algorithm that achieves a tight upper bound of $O(\sqrt{n})$, where n is again the number of obstacles. Recently, Berman *et al*. [22] gave a randomized algorithm that is $O(n^{4/9} \log n)$-competitive against any oblivious adversary. An interesting open problem is to develop improved randomized online algorithms.

Better competitive ratios can be achieved if the rectangles lie in an $n \times n$ square room and the robot has to reach the center of the room. For this problem, Bar-Eli *et al*. [15] gave tight upper and lower bounds of $\Theta(n \log n)$.

Further work on navigation has concentrated, for instance, on extending results to scenes with convex obstacles or to three-dimensional scenes [24,25].

Further online problems

There are many online problems that we have not addressed in this survey. *Metrical task systems*, introduced by Borodin *et al*. [27], can model a wide class of online problems. A metrical task system consists of a pair (S,d), where S is a set of n states and d is a cost matrix satisfying the triangle inequality. Entry $d(i,j)$ is the cost of changing from state i to state j. A task system must serve a sequence of *tasks* with low total cost. The cost of serving a task depends on the state of the system. Borodin *et al*. [27] gave a deterministic $(2n-1)$-competitive online algorithm. Recently, Bartal *et al*. [14] gave randomized algorithms achieving a polylogarithmic competitive ratio.

Online coloring and *online matching* are two classical online problems related to graph theory. In these problems, the vertices of a graph arrive online and must be colored or matched immediately. We refer the reader to [49,50,55,68] for some basic literature.

Further interesting online problems arise in the areas of financial games (e.g. [29,38], virtual circuit routing (e.g. [6,7,40]), Steiner tree construction (e.g. [23], or dynamic storage allocation (e.g. [54]).

Refinements of competitive analysis

Competitive analysis is a strong worst-case performance measure. For some problems, such as paging, the competitive ratios of online algorithms are much higher than the corresponding performance ratios observed in practice. For this reason, a recent line of research evaluated online algorithms on restricted classes of request sequences. In other words, the power of an adversary is limited.

In [26,45], competitive paging algorithms with *access graphs* are studied. In an access graph, each node represents a page in the memory system. Whenever a page p is requested, the next request can only be to a page that is adjacent to p in the access graph. Access graphs can model more realistic request sequences that exhibit locality of reference. It was shown that, using access graphs, it is possible to overcome some negative aspects of conventional competitive paging results [26,45].

With respect to online financial games, Raghavan [60] introduced a *statistical adversary*: The input generated by the adversary must satisfy certain statistical assumptions. In [29], Chou *et al*. developed further results in this model.

More generally, Koutsoupias and Papadimitriou [53] proposed the *diffuse adversary model*. An adversary must generate an input according to a probability distribution D that belongs to a class Δ of possible distributions known to the online algorithm.

References

[1] D. Achlioptas, M. Chrobak and J.Noga. Competitive analysis of randomized paging algorithms. In *Proc. Fourth Annual European Symp. on Algorithms (ESA)*, Springer LNCS, Vol. 1136, 419-430, 1996.

[2] S. Albers. Improved randomized on-line algorithms for the list update problem. In *Proc. 6th Annual ACM-SIAM Symp. on Discrete Algorithms*, 412-419, 1995.

[3] S. Albers. Better bounds for online scheduling. In *Proc. 29th Annual ACM Symp. in Theory of Computing*, 130-139, 1997.

[4] S. Albers and M. Henzinger. Exploring unknown environments. In *Proc. 29th Annual ACM Symp.\ in Theory of Computing*, 416-425, 1997.

[5] S. Albers, B. von Stengel and R. Werchner. A combined BIT and TIMESTAMP algorithm for the list update problem. *Information Processing Letters*, 56:135-139, 1995.

[6] J. Aspnes, Y. Azar A. Fiat, S. Plotkin and O. Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. In *Proc. 25th ACM Annual ACM Symp on the Theory of Computing*, 623-631, 1993.

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

[8] B. Awerbuch, Y. Bartal and A.Fiat. Competitive distributed file allocation. In *Proc. 25th Annual ACM Symp. on Theory of Computing*, 164-173, 1993.

[9] Y. Azar. On-line load balancing. Survey that will appear in a book on online algorithms, edited by A. Fiat and G. Woeginger, Springer Verlag.

[10] Y. Azar, A. Broder and A. Karlin. On-line load balancing. In *Proc. 36th IEEE Symp. on Foundations of Computer Science*, 218-225, 1992.

[11] Y. Azar, B. Kalyanasundaram, S. Plotkin, K. Pruhs and O. Waarts. Online load balancing of temporary tasks. In *Proc. Workshop on Algorithms and Data Structures*, Springer LNCS, 119-130, 1993.

[12] Y. Azar, J. Naor and R. Rom. The competitiveness of on-line assignments. In *Proc. of the 3th Annual ACM-SIAM Symp. on Discrete Algorithms*, 203-210, 1992.

[13] R.A. Baeza-Yates, J.C. Culberson and G.J.E. Rawlins. Searching in the plane. *Information and Computation*, 106:234-252, 1993.

[14] Y. Bartal, A. Blum, C. Burch and A. Tomkins. A polylog(n)-competitive algorithm for metrical task systems. In *Proc. 29th Annual ACM Symp. in Theory of Computing*, 711-719, 1997.

[15] E. Bar-Eli, P. Berman, A. Fiat and P. Yan. On-line navigation in a room. In *Proc. 3rd ACM-SIAM Symp. on Discrete Algorithms*, 237-249, 1992.

[16] Y. Bartal, M. Charikar and P. Indyk. On page migration and other relaxed task systems. In *Proc. of the 8th Annual ACM-SIAM Symp. on Discrete Algorithms*, 1997.

[17] Y. Bartal, A. Fiat, H. Karloff and R. Vohra. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences*, 51:359-366, 1995.

[18] Y. Bartal, A. Fiat and Y. Rabani. Competitive algorithms for distributed data management. In *Proc. 24th Annual ACM Symp. on Theory of Computing*, 39-50, 1992.

[19] L.A. Belady. A study of replacement algorithms for virtual storage computers. *IBM Systems Journal*, 5:78-101, 1966.

[20] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11:2-14,1994.

[21]J.L. Bentley, D.S. Sleator, R.E. Tarjan and V.K. Wei. A locally adaptive data compression scheme. *Communication of the ACM*, 29:320-330, 1986.

[22] P. Berman, A. Blum, A. Fiat, H. Karloff, A. Rosén and M. Saks. Randomized robot navigation algorithm. In *Proc. 4th Annual ACM-SIAM Symp. on Discrete Algorithms*, 74-84, 1996.

[23] P. Berman and C. Coulston. On-line algorithms for Steiner tree problems. In *Proc. 29th Annual ACM Symp. in Theory of Computing*, 344-353, 1997.

[24] P. Berman and M. Karpinski. The wall problem with convex obstacles. Manuscript.

[25] A. Blum, P. Raghavan and B. Schieber. Navigating in unfamiliar geometric terrain. In *Proc. 23th Annual ACM Symp.on Theory of Computing*, 494-504, 1991.

[26] A. Borodin, S. Irani, P. Raghavan and B. Schieber. Competitive paging with locality of reference. In *Proc. 23rd Annual ACM Symp. on Theory of Computing*, 249-259, 1991.

[27] A. Borodin, N. Linial and M. Saks. An optimal on-line algorithm for metrical task systems. *Journal of the ACM*, 39:745-763, 1992.

[28] M. Burrows and D.J. Wheeler. A block-sorting lossless data compression algorithm. DEC SRC Research Report 124, 1994.

[29] A. Chou, J. Cooperstock, R. El Yaniv, M. Klugerman and T. Leighton. The statistical adversary allows optimal money-making trading strategies. In *Proc. 6th Annual ACM-SIAM Symp. on Discrete Algorithms*, 467-476, 1995.

[30] M. Chrobak and L.L. Larmore. An optimal online algorithm for *k* servers on trees. *SIAM Journal on Computing*, 20:144-148, 1991.

[31]R. Cole. On the dynamic finger conjecture for splay trees. Part 2: Finger searching. Technical Report 472, Courant Institute, NYU, 1989.

[32] R. Cole. On the dynamic finger conjecture for splay trees. In *Proc. 22nd Annual ACM Symp. on Theory of Computing*, 8-17, 1990.

[33] R. Cole, B. Mishra, J. Schmidt, and A. Siegel. On the dynamic finger conjecture for splay trees. Part 1: Splay sorting log ***n***-block sequences. Technical Report 471, Courant Institute, NYU, 1989.

[34] D. Coppersmith, P. Doyle, P. Raghavan and M. Snir. Random walks on weighted graphs, and applications to on-line algorithms. *Journal of the ACM*, 1993.

[35] X. Deng, T. Kameda and C.H. Papadimitriou. How to learn an unknown environment. *Proc. 32nd Symp. on Foundations of Computer Science*, 298-303, 1991.

[36] X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *Proc. 31st Symp. on Foundations of Computer Science*, 356-361, 1990.

[37] G. Dudek, K. Romanik and S. Whitesides. Localizing a robot with minimum travel. In *Proc. 6th ACM-SIAM Symp. on Discrete Algorithms*, 437-446, 1995.

[38] R. El-Yaniv, A. Fiat, R.M. Karp and G. Turpin. Competitive analysis of financial games. In *Proc. 33rd Annual Symp. on Foundations of Computer Science*, 327-333, 1992.

[39] A. Fiat, R.M. Karp, L.A. McGeoch, D.D. Sleator and N.E. Young. Competitive paging algorithms. *Journal of Algorithms*, 12:685-699, 1991.

[40] J. Garay, I.S. Gopal, S. Kutten, Y. Mansour and M. Yung. Efficient online call control algorithms. In *Proc. 2nd Israel Symp. on Theory of Computing and Systems*, 285-293, 1993.

[41] R.L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563-1581, 1966.

[42] E.F. Grove. The Harmonic online *k*-server algorithm is competitive. In *Proc. of the 23rd Annual ACM Symp. on Theory of Computing*, 260-266, 1991.

[43] L. Guibas, R. Motwani and P. Raghavan. The robot localization problem in two dimensions. In *Proc. 3rd ACM-SIAM Symp. on Discrete Algorithms*, 259-268, 1992.

[44] F. Hoffmann, C. Icking, R. Klein and K.Kriegel. A competitive strategy for learning a polygon. *Proc. 8th ACM-SIAM Symp. on Discrete Algorithms*, 166-174, 1997.

[45] S. Irani, A.R. Karlin and S. Phillips. Strongly competitive algorithms for paging with locality of reference. In *Proc. 3rd Annual ACM-SIAM Symp. on Discrete Algorithms*, 228-236, 1992.

[46] D.R. Karger, S.J. Phillips and E. Torng. A better algorithm for an ancient scheduling problem. *Journal of Algorithms*, 20:400-430, 1996.

[47] A. Karlin, M. Manasse, L. Rudolph and D.D. Sleator. Competitive snoopy caching, *Algorithmica*, 3:79-119, 1988.

[48] R. Karp and P. Raghavan. From a personal communication cited in [62].

[49] R. Karp, U. Vazirani and V. Vazirani. An optimal algorithm for online bipartite matching. In *Proc. 22nd ACM Symp. on Theory of Computing*, 352-358, 1990.

[50] S. Khuller, S.G. Mitchell and V.V. Vazirani. On-line weighted bipartite matching. In *Proc. 18th International Colloquium on Automata, Languages and Programming (ICALP)*, Springer LNCS, Vol. 510, 728-738, 1991.

[51] J.M. Kleinberg. The localization problem for mobile robots. In *Proc. 35th IEEE Symp. on Foundations of Computer Science*, 521-531, 1994.

[52] E. Koutsoupias and C.H. Papadimitriou. On the *k*-server conjecture. *Journal of the ACM*, 42:971-983, 1995.

[53] E. Koutsoupias and C.H. Papadimitriou. Beyond competitive analysis. In *Proc. 34th Annual Symp. on Foundations of Computer Science*, 394-400, 1994.

[54] M. Luby, J. Naor and A. Orda. Tight bounds for dynamic storage allocation. In P*roc. 5th ACM-SIAM Symp. on Discrete Algorithms*, 724-732, 1994.

[55] L. Lovasz, M. Saks and M. Trotter. An online graph coloring algorithm with sublinear performance ratio. *Discrete Mathematics*, 75:319-325, 1989.

[56] M.S. Manasse, L.A. McGeoch and D.D. Sleator. Competitive algorithms for on-line problems. In *Proc. 20th Annual ACM Symp. on Theory of Computing*, 322-33, 1988.

[57] L.A. McGeoch and D.D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6:816-825, 1991.

[58] R. Motwani and P. Raghavan. Randomized Algorithms, Cambridge University Press, 1995.

[59] C.H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84:127-150, 1991.

[60] P. Raghavan. A statistical adversary for on-line algorithms. In *On-Line Algorithms*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 79-83, 1991.

[61] P. Raghavan and M. Snir. Memory versus randomization in on-line algorithms. In *Proc. 16th International Colloquium on Automata, Languages and Programming*, Springer LNCS, Vol. 372, 687-703,1989.

[62] N. Reingold, J. Westbrook and D.D. Sleator. Randomized competitive algorithms for the list update problem. *Algorithmica*, 11:15-32, 1994.

[63] K. Romanik and S. Schuierer. Optimal robot localization in trees. In *Proc. 12th Annual Symp. on Computational Geometry*, 264-273, 1996.

[64] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communication of the ACM*, 28:202-208, 1985.

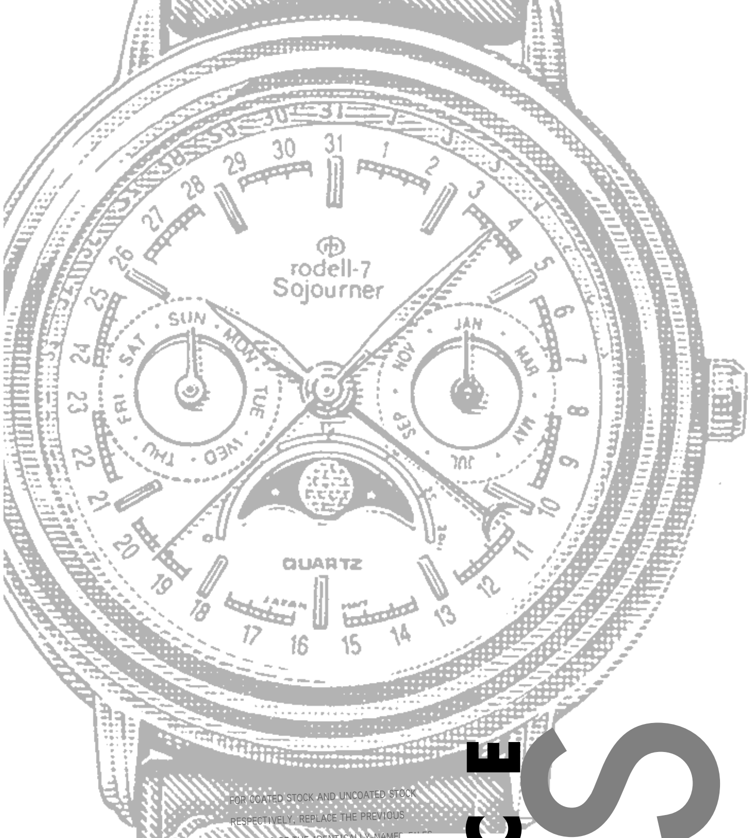
[65] D.D. Sleator and R.E. Tarjan. Self-adjusting binary search trees. *Journal of the ACM*, 32:652-686, 1985.

[66] R. E. Tarjan. Sequential access in splay trees takes linear time. *Combinatorica*, 5(4):367-378, 1985.

[67] B. Teia. A lower bound for randomized list update algorithms. *Information Processing Letters*, 47:5-9, 1993.

[68] S. Vishwanathan. Randomized online graph coloring. In *Proc. 31st Annual IEEE Symp. on Foundations of Computer Science*, 1990.

[69] J. Westbrook. Randomized algorithms for the multiprocessor page migration. *SIAM Journal on Computing*, 23:951-965, 1994.



CONFERENCE NOTES

▶ **HPSNO 97 High Performance Software for Nonlinear Optimization: Status and Perspectives**
Ischia, Italy
June 4-6, 1997

▶ **MPS at EURO/INFORMS**
Barcelona, Spain
July 14-17, 1997

▶ **NOAS '97 Nordic Operations Research Conference**
Copenhagen, Denmark
Aug. 15-16, 1997

▶ **XVI International Symposium on Mathematical Programming**
Lausanne, Switzerland
Aug. 24-29 1997

▶ **Algorithms and Experiments (ALEX98) Building Bridges Between Theory and Applications**
Trento, Italy
Feb. 9-11, 1998

▶ **ICM98**
Berlin, Germany
Aug. 18-27, 1998

CALL FOR PAPERS

NOAS '97

Nordic Operations Research Conference
University of Copenhagen
Denmark
August 15-16, 1997

NOAS is a biennial conference organized by the Nordic Operations Research Societies (Nordisk Operations Analyse Symposium). In 1997 the conference is hosted by the Danish Operations Research Society, DORS. The aim is to bring together researchers and practitioners to present and discuss operations research themes and to strengthen the contacts between people in the Nordic countries working in OR and related fields.

The conference is open to everyone interested in operations research and related subjects and is not restricted to participants from the Nordic countries.

The conference will be held at the University of Copenhagen (Campus of Natural Sciences), Copenhagen, Denmark, and will be preceeded by an informal get-together reception on the 14th of August. The campus of Natural Sciences is located centrally in Copenhagen, about a 15- to 20-minute walk from the medieval city centre.

The conference language is English.

All authors are requested to submit a paper/extended abstract of their talk. Papers should be typeset on one side of the paper only with wide margins and preferably not longer than 15 pages. The title page should give the following information: Title of paper, author(s), affiliation(s), and keywords. The papers will not be refereed and may subsequently be submitted for publication in appropriate journals.

Program:

Session I stream 1: Energy
Session I stream 2: Models and Solution techniques
Session II: Telecommunication/Energy
Session III stream 1: Linear Programming
Session III stream 2: Modelling
Session IV stream 1: Transportation
Session IV stream 2: Heuristics
Session V: Transportation
Session VI: Solution Techniques

Program Committee:

Jens Clausen, DORS (Chair); Snjolfur Olafsson, ICORS; Anders Eriksson, SORA; Margareta Soismaa, FORS; Arne Lokketangen, Norway

Organization Committee:

Claus C. Caroe, David Pisinger,
Jens Moberg Rygaard

Registration:

The registration fee includes conference proceedings, lunches, refreshments, get-together reception, conference dinner and evening meal on the 16th of August.

The general conference fee is 1900 DKR. Payment must be remitted in Danish Kroner - DKR. Checks should be made payable to NOAS'97, c/o DIKU, and sent to the address below together with the registration form. Payment by bank transfer should be made to NOAS'97, bank account no. 1521-260-01 in UNIbank, Lyngbyvej 20, DK-2100 Copenhagen, reg. no. 2113, SWIFT address UNIBDKKK, free from all bank charges. Please mark all payments clearly with the name of the participant.

Please use the information below or contact the conference secretariat for information on availability of hotel rooms.

Turistinformationen: Bernstorffsgade 1
(next to Central Railway Station and Tivoli
1577 Copenhagen V
Tel.: (+45) 33 11 13 25
(24-hour information service)
Information: (+45) 33 11 13 25
(Mon-Fri: 9-14; Sat: 10-14, Sun: Closed)
Hotel booking: (+45) 33 12 28 80 (same hours)

Conference Secretariat:

NOAS'97
c/o DIKU
Universitetsparken 1
DK-2100 Copenhagen
Denmark
E-mail: noas97@math.ku.dk
WWW:http://www.math.ku.dk/or/noas97/

Algorithms and Experiments (ALEX98)

Building Bridges Between Theory and Applications
Trento, Italy
February 9-11, 1998

The aim of this workshop is to provide a discussion forum for researchers and practitioners interested in the design, analysis and experimental testing of exact and heuristic algorithms.

In particular, the workshop will address methodological issues and significant case studies in the area of experimental analysis and tuning of algorithms, a subject whose importance is being recognized by a growing number of researchers in the CS community, both to assess the relevance or limitations of theoretical models and to create bridges toward applications in different domains.

The scientific program will include the presentation of invited talks and contributed research papers. We are interested in general experimentation methods, new algorithms developed through focussed experiments, and theoretical results motivated by experimental studies. More practical issues like standards of software development/documentation aimed at scientific research, procedures for testing algorithms via the web, and significant applications will also be considered.

We are aiming at a limited attendance, in order to have better contacts and exchange of ideas among the participants, both through the formal presentations and through discussions and brainstorming sessions in the relaxed environment offered by the local winter season.

The workshop is organized by the Computer Science Laboratory at the Department of Mathematics, University of Trento, Italy.

For more information contact:
http://rtm.science.unitn.it/alex98
email: alex98@rtm.science.unitn

Linear Programming
A Modern Integrated Analysis

R. Saigal

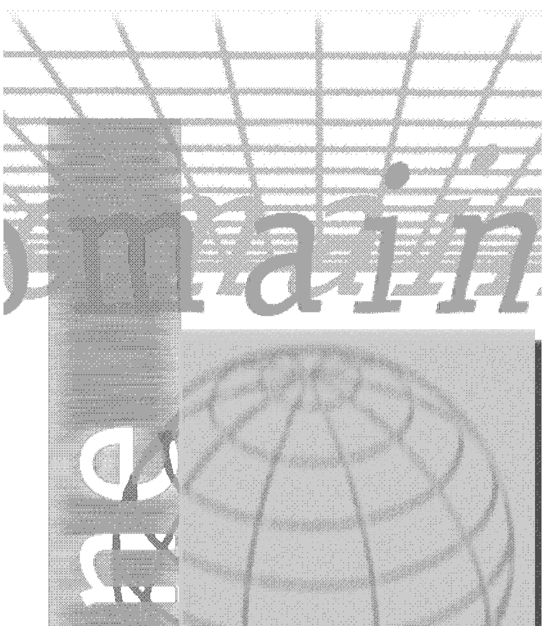
Kluwer Academic Publishers, Dordrecht, 1995

ISBN 0-7923-9622-7

As the research on interior point methods for linear programming has matured, the pedagogy of linear programming has been urged to cover not only the simplex method but also the new methods. However, this is not so easy in practice since the usual linear programming textbooks (with emphasis on the simplex method) do not provide the entire mathematical background of the recent advances. To overcome this difficulty, the publishing of new textbooks has flourished, especially in the last three years. This book of Romesh Saigal, written in a relatively early stage of this trend, will give several hints on introducing interior point methods into a course in linear programming.

The book consists of six chapters and an appendix. Chapters 1 through 3 present the mathematical tools and the fundamental results which are used in this book to analyze the simplex method and its variants (the author calls them *boundary methods*) and the interior point methods. A remarkable feature is that about 30 percent of this part is allocated to descriptions of real analysis and theory of nonlinear systems. In contrast to studying the simplex methods, this background is necessary for studying interior point methods, and hence teaching the methods often looks laborious. The appropriate summary of this book will be helpful to each reader who is interested in this subject.

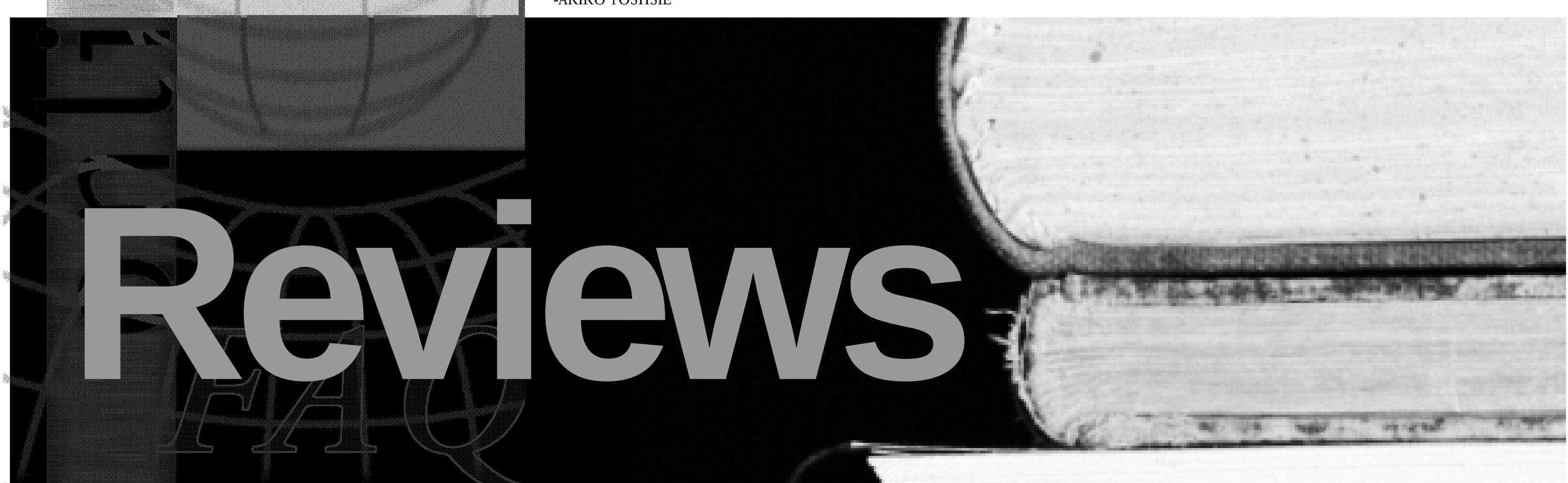
Chapters 4 and Chapter 5 deal with the boundary methods and the interior point methods, respectively. The table in Section 4.1 summarizes the differences among these methods and gives us a glimpse of a goal of this book: *This book presents both the boundary and the interior point methods in a unified manner* (quoted from Preface). By exhibiting a sufficient number of basic results in previous chapters, Saigal succeeds in presenting the boundary methods (including the primal and the dual simplex methods and the primal and the primal dual method) briefly but clearly in Chapter 4. On the other hand, the number of pages for describing the interior point methods is about six times as large as the one for the boundary methods. In particular, Saigal devotes more than 60 percent of this part to discussing the primal affine scaling method and its variants in detail. The clear explanation of these methods promotes the understanding of the mechanisms to solve degenerate problems and to attain superlinear and/or quadratic convergence. Polynomial time methods, i.e., path following methods using the predictor-corrector strategy and the projective transformation method developed by Karmarkar, are presented together with proofs for their polynomiality. However, reading these chapters may require considerable effort for some readers. The absence of any figures prevents beginners from having geometric intuition of the methods. Also, the various entities appearing in Chapter 5, have few descriptions of their meanings or roles in the analyses. Helpful comments from one with knowledge of the methods would be desirable for such readers.



Chapter 6 covers basic techniques for implementing both the boundary and the interior point methods. Several matrix factorization methods are presented. Among others, Saigal places the emphasis on the sparse and the partial Cholesky factorizations combined with the conjugate gradient method. Some instructive results on numerical experimentation of the methods are given in the appendix.

This book offers insight into recent developments in linear programming with a special interest in the study of affine scaling methods. Reading this book will be more pleasant for readers when comparing it with other books on interior point methods, some of which focus on the primal-dual interior point methods (based on the path following strategy) with different intentions.

-AKIKO YOSHSIE



Nondifferentiable and Two-level Mathematical Programming

K. Shimizu, Y. Ishizuka, and J.F. Bard

Kluwer Academic Publishers, Dordrecht, 1997

ISBN 0-7923-9821-1

As the title suggests, this book is concerned with nondifferentiable mathematical programming and two-level optimization problems. The emphasis is on presenting basic theoretical principles and on developing optimality conditions rather than on discussing algorithms, although a few computational approaches are briefly addressed. The book first discusses nondifferentiable nonlinear programming problems and characterizes directional derivatives and optimality conditions. This theory is then used to study two-level mathematical programs, where the presence of optimal value functions within the model renders them nondifferentiable.

The book contains 16 chapters. Chapter 1 introduces the different problems and applications that are discussed throughout the book, and Chapter 2 provides basic background material for differentiable and nondifferentiable nonlinear programming problems. Standard supporting and separating hyperplane results, the characterization of subdifferentials and generalized directional derivatives, and various theorems of the alternative are presented.

Chapter 3 deals with differentiable nonlinear programming problems. The Karush-Kuhn-Tucker (KKT) theory is developed for unconstrained as well as for constrained problems, along with saddle point duality theorems. Algorithmic approaches for both unconstrained and constrained problems, including Newton, quasi-Newton, conjugate gradient, penalty, and feasible directions methods are briefly addressed. A nice addition here is a discussion on multi-objective programs, including the concept of efficient solutions and related necessary and sufficient optimality conditions. Chapter 4 then addresses the extension of these concepts to nondifferentiable optimization problems. This chapter characterizes directional derivatives and develops KKT-type of optimality conditions for locally Lipschitz and quasi-differentiable cases. A very brief outline of subgradient optimization and bundle methods is also presented.

Chapter 5 deals with a specialization of these results to linear programming problems, focusing mainly on the simplex method and duality and sensitivity analysis results.

Chapter 6 begins to lay the groundwork for connecting the two parts of this book. Optimal-value functions that are parameterized by some variable set are introduced, and for these functions, continuity properties, KKT multiplier maps under suitable constraint qualifications, directional derivatives and generalized gradients are explored. A special case in which the constraint map does not depend on the parameters (the *nonparametric* case) is also treated.

Chapter 7 provides an introduction to two-level mathematical programming problems and Stackelberg leader-follower problems. For two-level nonlinear programming problems, optimality conditions are developed for the nonparametric case where the lower level constraints do not depend on the upper level decisions and for the parametric case where they do. For Stackelberg problems, optimality conditions are again developed for both cases where the lower level optimal solution map is differentiable or nondifferentiable. An application of bundle methods to solve this problem is described, and several applications to other problems such as minmax, satisfaction, two-level design, resource allocation, and approximation theory, among others, are presented.

Chapter 8 deals with decomposition methods for large-scale nonlinear programming problems that exhibit a block diagonal structure. Both primal decomposition and Lagrangian duality based decomposition methods are described.

Chapters 9 through 14 focus on the aforementioned applications, addressing, in turn, minmax problems, satisfaction optimization problems, two-level design problems, general resource allocation problems for decentralized systems, minmax multiobjective problems, and best approximation methods via Chebyshev norms. In each case, optimality conditions are developed for both the parametric and nonparametric cases, depending on whether or not the second level constraints are governed by decisions made in the first stage. Finally, Chapter 15 discusses the general Stackelberg leader-follower problem, and Chapter 16 specializes this discussion to the case of linear and convex function structures. For this latter instance, detailed algorithms are developed for linear and convex bilevel programming problems, including situations where the model incorporates certain discrete decision variables. The book concludes with a selected set of a few references that highlight the vast breadth of topics addressed in this book.

Overall, the book presents a nice basic, fundamental introduction to nondifferentiable and two-level optimization problems, along with related applications and possible solution approaches. The book is not intended to be used as a textbook. It contains no exercises and only a few illustrative examples. The audience addressed are mainly post graduate students and researchers who will find useful information here in beginning to study this vast and interesting topic.

-HANIF D. SHERALI

Introduction to Linear Optimization

D. Bertsimas and J.N. Tsitsiklis

Athena Scientific, P.O. Box 391, Belmont, MA 02178-9998, 1997

ISBN 1-886529-19-1

This new book on linear programming and closely related areas is published by Athena Scientific, which specializes in books written by M.I.T. faculty, based on courses taught there. It treats linear programming both extensively and thoroughly, while the related topics of linear network optimization, large scale optimization and integer programming receive concise treatments. The book is suitable for a first-year graduate course on linear optimization, addressing doctoral and more mathematically inclined masters level students in Operations Research, Computer Science, Applied Mathematics and Management Science. The book certainly deserves to be on the shelves of all researchers who work directly in mathematical programming and those who apply the techniques. The true merit of this book, however, lies in its pedagogical quailities which are so impressive that I have decided to adopt it for a course on linear programming that I am scheduled to teach in the coming fall semester. To follow is an overview of the material covered in the book.

In the introductory Chapter 1, some variants of LP models are defined and standard reductions are given. The chapter also contains interesting "real world" examples of LP and other models, the "graphical method" for two-variable LP problems, the requisite linear algebra background, as well as a quick discussion on arithmetic complexity (complexity issues are given a more thorough treatment in later chapters). Chapter 2 provides some fundamental geometric insight behind LP. Basic notions concerning polyhedra and convex sets, such as extreme points and their existence, degeneracy and some underlying geometric insight, certain optimality issues in LP, and Fourier-Motzkin elimination are discussed.

The simplex method is systematically developed in Chapter 3. After deriving the optimality conditions, the mechanics of the simplex method are developed, and the revised simplex as well as the tableau form implementations are discussed. Anti-cycling rules, two-phase (along with an oft ignored aspect of driving the artificial variables out of the basis) and "Big-M" methods are then presented followed by some geometric insight into the primal simplex method, using what the authors call "column geometry." The chapter ends with a discussion of worst case and average case complexity of the simplex method along with a few words on the diameters of polyhedra. The only missing aspect in the coverage of the simplex method is the upper bounded simplex method.

LP duality theory is the subject of Chapter 4. The authors first derive the LP dual using the notions of Lagrangian duality which, in my opinion, is highly pedagogically efficient. Then, the weak and strong duality theorems are proved using the workings of the simplex method. And finally, an alternate derivation of duality via convex analysis is given. The development here is supported with many geometric and intuitive explanations and interpretations.

In Chapter 5, sensitivity analysis is presented. The local sensitivity analysis is fairly standard material with the exception of sensitivity with respect to a coefficient of the constraint matrix in a basic column. In the latter part, what the authors call "global sensitivity" is presented which deals with the optimal objective value as a function of either the right hand side vector or the cost vector of the objective function. The chapter ends with a quick introduction to parametric optimization. The topics of large-scale optimization such as delayed column generation, cutting plane methods, Dantzig-Wolfe method and Bender's decomposition are discussed in Chapter 6. The coverage here is rather brisk but is sufficient for a good exposure of these useful techniques to the student.

Chapter 7 deals with network flow problems. After introducing some graph theoretic notation, the various types of network flow models are stated. Then, the network simplex method for the uncapacitated case is presented, with the capacitated version being specified as an extension. In my opinion, it would have been better to treat the more general capacitated case first and then specialize it to the transshipment problem. The other topics include the negative cost cycle algorithm, Ford-Fulkerson algorithm for the maximum flow problem, the max flow-min cut theorem, dual-ascent methods, auction algorithm for the assignment problem, shortest path and minimum spanning tree problems. In trying to cover too many topics in one chapter, I feel that the authors have somewhat compromised the clarity of exposition in this chapter.

In Chapters 8 and 9 polynomial time algorithms for LP are discussed. These two chapters, along with Chapter 12, alone are a good enough reason for one to buy this book. The material here is beautifully written and wonderfully presented. Chapter 8 deals with the often disregarded ellipsoid method. The important fact that the ellipsoid method can be used to solve problems with exponentially many constraints (as long as we have an efficient separation oracle) is well emphasized. In Chapter 9 on interior point methods, three broad classes of interior point algorithms, namely, affine scaling, potential reduction, and path following algorithms, are presented and analyzed. The material in these two chapters is concise yet thorough, involved yet easy to follow, and it leaves the reader with a clear understanding of the key ideas behind polynomial time algorithms for LP.

Integer programming formulations and methods are discussed in Chapters 10 and 11. The first of these chapters has some standard IP models, as well as models with exponentially many constraints. In Chapter 11, the authors begin by discussing the Gomory cutting plane method (but they omit the rather nice convergence proof, apparently owing to space limitations). Then branch-and-bound as well as branch-and-cut techniques, and the dynamic programming algorithm for the Knapsack problem are briefly presented. Lagrangian duality as it pertains to IP is presented, and assorted topics such as approximation algorithms, local search methods, and simulated annealing are discussed. Then, out of the blue, a section on rigorous notions of complexity classes (P, NP, NP-Complete etc.) appears, perhaps owing to the lack of a better place within the structure of the book.

The final chapter titled, "The art in linear optimization", is unique to this book. It is designed to turn the pretty theory of the first 11 chapters magically into practical problem solving ability. It contains brief discussions of modeling languages, optimization software libraries, and tricky aspects such as preprocessing, choice of algorithms, effective heuristics, and other practical tidbits which make large-scale real-world problem solving more effective.

I will now offer some general comments about the book. An innovative technique used by the authors is to pose as exercise problems, at the end of each chapter, some interesting topics that are vital but can be derived fairly easily from the tools presented in that chapter. This expands the coverage of the material without making the book too voluminous. For example, one of the exercises at the end of Chapter 2 is the perennially useful Caratheodory's theorem. Another example is the Clark's theorem (which states that if at least one of the primal or the dual problems is feasible, then at least one of the two feasible regions is unbounded). Clearly, these exercises make for challenging homework problems for the ambitious teacher within us.

Throughout the book, the authors make serious efforts to give geometric and intuitive explanations of various algebraic concepts, and they are widely successful in this effort. An example of this is witnessed in chapter 4, where the authors provide a visualization tool that helps one picture dual feasible bases and solutions in the primal space. Many of these explanations and insights are the things that seasoned math programmers grasp over the course of their research careers. The authors' quest for completeness of presentation is easily noticed at many places in the text and is very appreciable.

Although at times the phraseology takes the tone of a research paper and in some places the material feels dense (mainly because a long list of topics is covered), the overall writing style is pleasant and to-the-point. The chapter-wise organization is nearly ideal, while the arrangement of sections within certain chapters may be reshuffled by the individual instructor to suit her or his own teaching style.

In conclusion, this is an outstanding textbook that presents linear optimization in a truly modern and up-to-date light. One reading of this book is sufficient to appreciate the tremendous amount of quality effort that the authors have put into the writing, and I strongly recommend it to all teachers, researchers and practitioners of mathematical programming.

-MOTAKURI V. RAMANA

Journals

Mathematical Programming

Vol. 77, No. 1

E. Schweitzer, ***A Gaussian upper bound for Gaussian multi-stage stochastic linear programs.***

P.T. Thach, ***On the degree and separability of nonconvexity and applications to optimization problems.***

R. Weismantel, ***On the 0/1 knapsack polytope.***

Ph.L. Toint, ***Non-monotone trust-region algorithms for nonlinear optimization subject to convex constraints.***

Vol. 77, No. 2

M. Overton, ***Semidefinite Programming.***

F. Alizadeh, ***Complementarity and nondegeneracy in semidefinite programming.***

M.V. Ramana, ***An exact duality theory for semidefinite programming and its complexity implications.***

P. Gahinet, ***The Projective Method for solving linear matrix inequalities.***

A. Nemirovski, ***The long-step method of analytic centers for fractional problems.***

M. Laurent, ***Connections between semidefinite relaxations of the max-cut and stable set problems.***

M. Mesbahi, ***A cone programming approach to the bilinear matrix inequality problem and its geometry.***

F. Rendl, ***A semidefinite framework for trust region subproblems with applications to large scale minimization.***

A. Shapiro, ***First and second order analysis of nonlinear semidefinite programs.***

Vol. 77, No. 3

D. Bertsimas, ***On the worst case complexity of potential reduction algorithms for linear programming.***

H. Hundal, ***Two generalizations of Dykstra's cyclic projections algorithm.***

B. Sturmfels, ***Variation of cost functions in integer programming.***

E. Cheng, ***Wheel inequalities for stable set polytopes.***

APPLICATION FOR MEMBERSHIP

I wish to enroll as a member of the Society.

My subscription is for my personal use and not for the benefit of any library or institution.

☐ *I will pay my membership dues on receipt of your invoice.*

☐ *I wish to pay by creditcard (Master/Euro or Visa).*

CREDITCARD NUMBER: ______________________ EXPIRY DATE: ______________________

FAMILY NAME: ______________________

MAILING ADDRESS: ______________________

____

TEL.NO.: ______________________ TELEFAX: ______________________

E-MAIL: ______________________

SIGNATURE ______________________

Mail to:

The Mathematical Programming Society, Inc.
c/o International Statistical Institute
428 Prinses Beatrixlaan
2270 AZ Voorburg
The Netherlands

Cheques or money orders should be made payable to The Mathematical Programming Society, Inc., in one of the currencies listed below. Dues for 1997, including subscription to the journal *Mathematical Programming*, are Dfl.110.00 (or \$65.00 or DM100.00 or £40.00 or FF330.00 or Sw.Fr.80.00).
Student applications: Dues are one-half the above rates. Have a faculty member verify your student status and send application with dues to above address.

Faculty verifying status

____

institution

____

O P T I M A
MATHEMATICAL PROGRAMMING SOCIETY

UNIVERSITY OF
FLORIDA

Center for Applied Optimization
371 Weil Hall
PO Box 116595
Gainesville FL 32611-6595 USA

FIRST CLASS MAIL

Donald W. Hearn, EDITOR
hearn@ise.ufl.edu

Karen Aardal, FEATURES EDITOR
Utrecht University
Department of Computer Science
P.O. Box 80089
3508 TB Utrecht
The Netherlands
aardal@cs.ruu.nl

Faiz Al-Khayyal, SOFTWARE & COMPUTATION EDITOR
Georgia Tech
Industrial and Systems Engineering
Atlanta, GA 30332-0205
faiz@isye.gatech.edu

Dolf Talman, BOOK REVIEW EDITOR
Department of Econometrics
Tilburg University
P.O. Box 90153
5000 LE Tilburg
The Netherlands
talman@kub.nl

Elsa Drake, DESIGNER

PUBLISHED BY THE
MATHEMATICAL PROGRAMMING SOCIETY &
GATOREngineering® PUBLICATION SERVICES
UNIVERSITY OF FLORIDA

Journal contents are subject to change by the publisher.

gallimaufry

OPTIMA now has a web site under construction. The address is http://www.ise.ufl.edu/~optima. **Karen Aardal** will assume the role of OPTIMA Editor starting with No. 55. She has been Features Editor since 1994... **Don Hearn**, editor since OPTIMA's inception in 1980, became Chair of the Industrial & Systems Engineering Department, University of Florida, on May 9. A special thanks to **Faiz Al-Khayyal** and **Dolf Talman** for their work as associate editors. Publication and distribution will continue from the University of Florida, with **Elsa Drake** as designer. Deadline for the next issue is Sept 15, 1997.