Theoretical Computer Science 568 (2015) 28–48

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Relative interval analysis of paging algorithms on access graphs ☆

Joan Boyar [a], Sushmita Gupta [b], Kim S. Larsen [a,*]

[a] Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark
[b] Graduate School of Informatics, Kyoto University, Yoshida Honmachi, Sakyo-ku, Kyoto 606-8501, Japan

ARTICLE INFO

Article history:
Received 14 January 2014
Received in revised form 13 September 2014
Accepted 29 November 2014
Available online 5 December 2014
Communicated by G. Ausiello

Keywords:
Online algorithms
Paging
Access graphs
Relative interval analysis

ABSTRACT

Access graphs, which have been used previously in connection to competitive analysis and relative worst order analysis to model locality of reference in paging, are considered in connection with relative interval analysis. The algorithms LRU, FIFO, FWF, and FAR are compared using the path, star, and cycle access graphs. In this model, some of the results obtained are not surprising. However, although LRU is found to be strictly better than FIFO on paths, it has worse performance on stars, cycles, and complete graphs, in this model. We solve an open question from Dorrigiv et al. (2009) [13], obtaining tight bounds on the relationship between LRU and FIFO with relative interval analysis.

1. Introduction

The paging problem is the problem of maintaining a subset of a potentially very large set of pages from memory in a significantly smaller cache. This problem, originating in the design of operating systems, has received much attention in online algorithms research, giving rise to many algorithms and techniques for analyzing them. This attention is largely due to the importance of the problem combined with the dissatisfaction with the results obtained using standard competitive analysis, which is unable to distinguish between many paging algorithms of varying quality.

The paging problem considered in online algorithms is the following: We have a memory that can hold N pages and a cache that can hold $k < N$. A paging algorithm must process a sequence of requests. Each request is a reference to a page in memory, which may or may not be present in the cache. If it is present, this is referred to as a "hit" and otherwise we have a "fault". The paging algorithm must bring the requested page into cache if it is not already there. Since the cache is smaller than the memory, this means that when a page not currently present in cache must be brought in, some other page must be evicted. The only decision a paging algorithm can make is which page to evict. For this reason, paging algorithms are also sometimes referred to as eviction strategies. The problem is online, meaning that when the paging algorithm processes a given request, it has no information about any future requests, and the objective is to minimize the total number of faults.

☆ A preliminary version of this paper appeared in the proceedings of the Thirteenth Algorithms and Data Structures Symposium, 2013. Supported in part by the Danish Council for Independent Research (FI-2146-09-0073 and DFF-1323-00247) and the Villum Foundation (VKR023219). Part of this work was carried out while the first and third authors were visiting the University of Waterloo.

* Corresponding author.
E-mail addresses: joan@imada.sdu.dk (J. Boyar), sushmita.gupta@gmail.com (S. Gupta), kslarsen@imada.sdu.dk (K.S. Larsen).

http://dx.doi.org/10.1016/j.tcs.2014.11.035
0304-3975/

We make the common assumption that the cache is empty to start with, so no evictions have to be carried out for the first k distinct page requests.

1.1. Paging algorithms

Many different paging algorithms have been considered in the literature, many of which can be found in [3,12]. Among the best known are LRU (least-recently-used), which always evicts the least recently used page, and FIFO (first-in-first-out), which evicts pages in the order they entered the cache. We also consider a known bad algorithm, FWF (flush-when-full), which is often used for reference, since quality measures ought to be able to determine at the very least that it is worse than the other algorithms. If FWF encounters a fault with a full cache, it empties its cache, and brings the new page in.

Finally, we consider a more involved algorithm, FAR^G [4], which works with respect to a known access graph, G. The access graph is an undirected graph with pages as vertices, which is used to limit the input sequences that are considered. The request sequence can start with any page, but for any point in the request sequence, the next request, if not identical to the previous, must be to a page connected to the current request by an edge in the access graph, i.e., the pages are adjacent. The intention is to provide a model for locality of reference.

The access graph is fixed before the request sequence starts and FAR^G uses this graph as well as marks that it administers on the pages in cache to make its decisions as follows: Whenever a page is requested, it is marked. On a fault, where it is necessary to evict a page, it always evicts an unmarked page. If all pages are marked in such a situation, FAR^G first unmarks all pages. The unmarked page it chooses to evict is the one farthest from any marked page in the access graph G. For breaking possible ties, when more than one unmarked page in cache has the maximal distance to marked pages, we use the LRU strategy in this paper, i.e., evicting the least recently used among these pages.

Other algorithms based on access graphs were investigated in [15–17].

1.2. Separation results via performance measures

Comparing the behavior of paging algorithms under various assumptions has been a topic for much research. The most standard measure of quality of an online algorithm, competitive analysis [21,18], cannot directly distinguish between most of them. The competitive ratio is inspired by the approximation ratio and is defined as follows: Letting $\text{ALG}(I)$ denote the cost (number of faults) of running an algorithm ALG on the input sequence I, ALG is c-competitive if there exists a constant b such that $\text{ALG}(I) \leq c\,\text{OPT}(I) + b$ for all I. Here OPT is an optimal offline algorithm. Thus, OPT knows the entire input sequence in advance, and its performance is a lower bound on how well any online algorithm can perform. The competitive ratio is the infimum over all c for which the algorithm is c-competitive. Competitive analysis deems LRU, FIFO, and FWF equivalent, with a competitive ratio of k [21,18], where k denotes the size of the cache.

Other measures, such as bijective/average analysis [2], relative worst order analysis [5,6], and relative interval analysis [13] can be used to obtain more separations. For the discussions here and later in the introduction, we briefly define these measures without introducing extra notation. More mathematically-based definitions, variations, and refinements of the measures can be found in the respective papers.

As opposed to competitive analysis, bijective analysis is based on a direct comparison between two algorithms, ALG and ALG′, rather than via a comparison to OPT. ALG is at least as good as ALG′ if for all n, when considering all possible different input sequences of length n, there exists a bijection σ on this set such that for all input sequences I of length n, $\text{ALG}(I) \leq \text{ALG}'(\sigma(I))$. Average analysis simply compares the average performance of the algorithms, again separately for each input length.

In relative worst order analysis, comparisons are also between two algorithms directly, but based on a partition of all possible input sequences. Using the notation from above, ALG is at least as good as ALG′ if for all input sequences I, $\text{ALG}_W(I) \leq \text{ALG}'_W(I)$, where $\text{ALG}_W(I)$ denotes ALG's performance on the permutation of I that gives the worst result (most faults).

Relative interval analysis considers the interval spanned by the fraction $\frac{\text{ALG}(I)-\text{ALG}'(I)}{n}$ over all input sequences I of length n. ALG dominates ALG′ if the right-most point of the interval is at most zero.

In contrast to competitive analysis, bijective, average, relative worst, and relative interval analysis all establish that LRU and FIFO are better than FWF, and also that look-ahead helps. Here look-ahead means that the strict online constraint that no information about the future is available is weakened, and an algorithm is allowed to see the l next requests for some constant l.

Since LRU performs better than FIFO in some practical situations [23], there has been considerable effort to explain this. To our knowledge, no techniques have been able to separate LRU and FIFO, without adding some modelling of locality of reference. In [20,24], there are proofs of optimality for LRU against a diffuse adversary, but as far as we can determine, no proof of non-optimality for FIFO.

One explanation of the difficulty in separating LRU and FIFO without considering locality of reference is as follows: If one considers all sequences of length n for any n, bijective/average analysis shows that their average number of faults on these sequences is identical [2], since both are demand paging algorithms.