On the Partitioning Algorithm

Béla Csaba

The *paging problem* is defined as follows. We have a two-level memory system with k pages of fast memory, and $n-k$ pages of slow memory. Repeatedly a request to a page appears. This request should be satisfied by moving the page to fast memory, if it is in slow memory, i.e., a page fault occurs. In this case a page must be evicted from fast memory to make room for the new, recently requested one. The paging problem is to decide which page is to be evicted.

There is a simple optimum paging algorithm, called MIN, if we know the whole request sequence in advance, in the *off-line* case. It is more practical to consider the *on-line* paging, when the algorithm has to decide immediately after a page request, without knowing what the future requests will be.

For comparing two paging algorithms the *competitive ratio* is used. This measure of performance of an on-line algorithm was introduced by Sleator and Tarjan (see [ST]). Fix any starting configuration of the pages, and denote by $opt(\sigma)$ the optimum number of page faults on request sequence σ, in other words, the optimum cost of σ. The competitive ratio of the on-line algorithm $\mathcal{A}$ is c, if there is a constant M such that on every request sequence σ the cost incurred by $\mathcal{A}$, $\mathcal{A}(\sigma)$ is at most $c\,opt(\sigma)+M$. It was shown (see [ST]) that no on-line algorithm can have a competitive ratio less than k. $LRU, FIFO$ and a large number of other on-line algorithms are known to be k-competitive.

As it happens frequently, one may expect a better performance in the randomized case. A randomized on-line algorithm $\mathcal{R}$ is c-competitive, if there is a constant M such that on every request sequence σ $E[\mathcal{R}(\sigma)]$ is at most $c\,opt(\sigma)+M$, where $E[\mathcal{R}(\sigma)]$ denotes the expected cost incurred by $\mathcal{R}$ on σ. It was proved (see [FKLMSY]) that $H_k = 1+\frac{1}{2}+\cdots+\frac{1}{k}$ is a lower bound for the randomized competitiveness of an on-line paging algorithm. There is a simple, elegant algorithm, which has randomized competitive ratio $2\,H_k$. On the other hand, the only known optimal randomized algorithm, the *partitioning algorithm* has a much more complicated description.

One of our results is that we give another description of the partitioning algorithm via another approach. This is done by analyzing the optimal satisfactions of request sequences. We construct a graph, and the random walks on this graph correspond to the random choices done by the partitioning algorithm. Besides, we prove that the algorithm is optimal, k-competitive in the deterministic case, and $(k-l)$-competitive having an *l-strong lookahead*, where $k-l>1$. This means that the on-line algorithm not only knows the present request and the previous ones, but has access to the first l element set of the future requests. No on-line algorithm can have less competitive ratio than $k-l$ with l-strong lookahead (see [A]), thus, the partitioning algorithm is optimal in this sense, too.

References

[A] Albers, S., *The influence of lookahead in competitive paging algorithms*, Proceedings of the 1st Annual European Symposium on Algorithms, 1993, pp. 1-12

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

[ST] Sleator, D. D., Tarjan, R. E., *Amortized efficiency of list update and paging rules*, Comm. of the ACM, February 1985, pp. 202-208