Lecture 13

Online Algorithms

- The ski problem: buy a new pair of skis or rent a pair each time (without knowing the future)?
- An answer: rent skis until the cumulative cost of renting first exceeds the cost of purchasing a new pair, and then purchase a pair at that point.
- With this strategy our cost never exceeds twice the minimum possible cost.

An algorithm is *online* if it has to respond immediately as each request is coming, without knowledge of future requests.

An important measure of performance for an online algorithm is the worst-case ratio of its performance to that of the optimal algorithm.

We say that an online algorithm A is c-competitive if for all reference strings σ,

$$C_A(\sigma) \leq c \cdot C_{OPT}(\sigma) + b.$$

The *competitive ratio* of A is the smallest c such that A is c-competitive. An online algorithm is *strongly* competitive if it achieves the smallest possible competitive ratio.

Competitive Analysis

- Page replacement [ST,PMC]
- Maintaining linear lists [ST]
- Snoopy caching [KMR,KMM]
- The k-server problem [MMS,FRR]
- Currency exchange, mortgage financing [EFKT]

Page Replacement

Two-level memory: fast memory and slow memory. e.g., main memory and auxiliary memory, cache and main memory.

- hit: a referenced block (page) is in fast memory
- miss (page fault): not in fast memory

A page replacement algorithm is that of deciding which page to remove from fast memory to make room for other pages to be brought in.

Memory management

- Fixed-space memory management: Each process has a fixed number of memory space. Count page faults.
- Variable-space memory management: Memory space can vary. Count page faults and keeping costs.

Optimal Algorithms

- Fixed-space: MIN due to Belady which removes the page that will remain unused for the furthest time into the future.
- Variable-space: VMIN due to Prieve and Fabry. Given the cost R of a page fault and the cost U of keeping one page in memory for one reference time, VMIN removes the page just referenced if and only if the page will not be referenced again for more than $\frac{R}{U}$ time units.
- Unrealizable, since optimal algorithms need to know future page references,
- Valuable as a benchmark for online algorithms.

Outline

Fixed space [ST]

- Lower bound for deterministic algorithms
- LRU and FIFO are strongly competitive

Variable space [PMC]

- Competitive ratio of WS
- Matching lower bound for deterministic algorithms
- Randomized algorithm SR
- Matching lower bound for randomized algorithms

- L.A. Belady, A study of replacement algorithms for a virtual storage computer, IBM Syst. J. 9 (1970), 78–117.
- P.J. Denning, The working set model for program behavior, Comm. ACM 11 (1968), 323–333.
- K. Park, S.L. Min and Y. Cho, The working set algorithm has competitive ratio less than two, Inform. Process. Lett. 63 (1997), 183–188.
- B.G. Prieve and R.S. Fabry, VMIN - an optimal variable-space page replacment algorithm, Comm. ACM 19, 5 (1976), 295–297.
- D.D. Sleator and R.E. Tarjan, Amortized efficiency of list update and paging rules, Comm. ACM 28 (1985), 202–208.

Fixed Space

Let k be the number of page frames in main memory (k is a constant). Count page faults.

Theorem 1 *The competitive ratio of any deterministic online algorithm A is at least k.*

Proof. We will construct a reference string on which A makes k faults and MIN makes only one fault.

Let p be a page not in A's or MIN's memory. Let S be the set of $k+1$ pages consisting of k pages in MIN's memory and p.

The first reference is to p; both A and MIN make a page fault. Each of the next $k-1$ references is to a page of S not currently in A's memory. Of these $k-1$ references, A faults every time. Since MIN keeps all pages for these $k-1$ references, MIN makes no page faults. This construction can be repeated. □

LRU and FIFO

- Least recently used (LRU): remove the page whose most recent reference was earliest.
- First-in, first-out (FIFO): remove the page that has been in memory longest.

Theorem 2 *For any reference string σ,*

$$C_{LRU}(\sigma) \leq kC_{MIN}(\sigma) + k.$$

Proof. After the first reference, LRU and MIN always have at least one page in common. Consider a substring s of σ during which LRU faults k times. Let p be the page referenced just before s. (Thus both LRU and MIN have p in memory.) We show that MIN faults at least once during s.

1. If LRU faults on the same page p' twice during s, then s must contain at least $k+1$ different pages. (To remove p', k other pages must be referenced.)
2. If LRU faults on p during s, then s also contains at least $k+1$ different pages.
3. Otherwise (i.e., LRU faults on k different pages, none of them p), one of the k pages cannot be in MIN's memory at the beginning of s because p is in.

Partition σ into $s_0, s_1, \ldots, s_t$ such that each of $s_1, \ldots, s_t$ contains exactly k faults by LRU and s_0 contains at most k faults by LRU. On $s_1, \ldots, s_t$ the ratio is at most k. During s_0, MIN may not fault. □

Same for FIFO. LRU and FIFO are strongly competitive. However LRU performs better than FIFO, considering locality of reference.

Variable Space

Let $\sigma = (p_1, p_2, \ldots, p_n)$ be a reference string and let t_i, $1 \leq i \leq n$, be the time p_i is referenced such that $t_1 < t_2 < \cdots < t_n$. We assume that t_i's are integers since the system is synchronized by the system clock.

The optimal variable-space page replacement algorithm VMIN keeps a referenced page if the time to its next reference is less than or equal to $\lfloor R/U \rfloor$ and removes it otherwise.

Working Set (WS): The working set algorithm [De1,De2] keeps a page in memory for T time units after it is referenced. T is called the window size of the working set algorithm.

Theorem 3 *Let* $\alpha = \lfloor \frac{R}{U} \rfloor \frac{U}{R}$ *and* $\beta = \frac{R-U}{\lfloor R/U \rfloor U}$. *The competitive ratio of the working set algorithm is* $1+\alpha$ *when* $T = \lfloor R/U \rfloor$ *and* $1+\beta$ *when* $T = \lfloor R/U \rfloor - 1$, *and therefore it is* $\min(1+\alpha, 1+\beta)$, *which is always less than 2.*

Proof. Since the same analysis applies to each referenced page, the problem reduces to the case of one page. Consider repeated references of a page. The worst ratio is obtained when the page is referenced every $T+1$ time units. Then WS pays $TU+R$ for every reference.

If $T \geq \lfloor R/U \rfloor$, VMIN pays R for every reference and thus the ratio is $\frac{TU+R}{R}$. If $T \leq \lfloor R/U \rfloor - 1$, VMIN pays $R + (n-1)(TU+U)$ for n references. Since $R \geq \lfloor R/U \rfloor U \geq TU + U$ in this case, the ratio is at most $\frac{TU+R}{TU+U}$.

Hence

$$C_{WS}(\sigma) \leq \frac{TU+R}{\min(R, TU+U)} C_{VMIN}(\sigma)$$

for any reference string σ.

Since we have freedom to choose window size T, we choose T which minimizes the ratio. For $T \geq \lfloor R/U \rfloor$, the minimum ratio is $1 + \lfloor R/U \rfloor \frac{U}{R}$ when $T = \lfloor R/U \rfloor$. For $T \leq \lfloor R/U \rfloor - 1$, it is $1 + \frac{R-U}{\lfloor R/U \rfloor U}$ when $T = \lfloor R/U \rfloor - 1$. Therefore, the competitive ratio of the working set algorithm is $\min(1+\alpha, 1+\beta)$.

Since $\lfloor R/U \rfloor \leq R/U$, we have $0 < \alpha \leq 1$. Since $\lfloor R/U \rfloor > \frac{R}{U} - 1$, we have $0 < \beta < 1$. Hence the competitive ratio of the working set algorithm is always less than 2. □

Compare $\alpha = \lfloor \frac{R}{U} \rfloor \frac{U}{R}$ and $\beta = \frac{R-U}{\lfloor R/U \rfloor U}$ to see which one gives the best ratio of the working set algorithm for given costs R and U.

Let $R/U = k + a$ for integer k and $0 \leq a < 1$. Then

$$\alpha - \beta = \frac{k}{k+a} - \frac{k+a-1}{k} = \frac{(1-2a)k + a(1-a)}{k(k+a)}.$$

- Since $0 \leq a < 1$, $a(1-a) \geq 0$. Hence if $a \leq 0.5$, we have $\alpha > \beta$.
- If $a > 0.5$, then $\alpha < \beta$ if and only if $k > \frac{a(1-a)}{2a-1}$. Note that $\frac{a(1-a)}{2a-1} = 1.2$ for $a = 0.6$ and it decreases as a increases. Since R is much larger than U in practice, we have $\alpha < \beta$ for most values of $a > 0.5$.

Therefore, $1 + \beta$ is the best ratio of the working set algorithm for $a \leq 0.5$ and $1 + \alpha$ is the best ratio for most values of $a > 0.5$. In particular, if U divides R, the best ratio is $2 - \frac{U}{R}$ when the window size T is $\frac{R}{U} - 1$.

Lower Bound

Theorem 4 *For any online algorithm A, there are arbitrarily long strings σ such that*

$$C_A(\sigma) \geq c \cdot C_{VMIN}(\sigma) - b,$$

where $c = \min(1 + \alpha, 1 + \beta)$ and $b = (c-1)R$.

Proof. We construct a string that consists of one page p. For the first reference, both A and VMIN cost R. The string is constructed in such a way that $C_A \geq c \cdot C_{VMIN}$ for each reference.

Suppose that page p is referenced at time t and A keeps p for x time units and removes it. If $x \leq \lfloor 2R/U \rfloor$, the next reference is at time $t + x + 1$. otherwise, the next reference is at time $t + \lfloor 2R/U \rfloor + 1$. $\square$

Randomized Algorithms

The definition of competitiveness for randomized algorithms is as follows [BBK,FKL]. Consider a randomized online algorithm B. Given a reference string σ, the cost of B is now a random variable, which is denoted by $C_B(\sigma)$. Thus the average of $C_B(\sigma)$ over all the random choices that B makes while processing σ is compared with the cost of the optimal algorithm.

We say that B is c-competitive if for every reference string σ,

$$E[C_B(\sigma)] \leq c \cdot C_{VMIN}(\sigma) + b$$

for some constant b. The competitive ratio of B is the smallest c such that B is c-competitive.

Strongly-Randomized

Randomized algorithm SR keeps a referenced page for i time units with probability r_i. We will determine r_i so as to minimize the competitive ratio of algorithm SR.

Suppose that a page p is referenced at time t and the next reference to p is at time $t + x$. Let s_x denote the time interval $(t..t + x]$. The expected cost of algorithm SR for s_x is

$$E[C_{SR}(s_x)] = \sum_{0 \leq i \leq x-1} r_i(R + iU) + (1 - \sum_{0 \leq i \leq x-1} r_i)xU.$$

The cost of the optimal algorithm VMIN for s_x is xU if $x \leq \lfloor R/U \rfloor$; it is R if $x > \lfloor R/U \rfloor$.

We choose r_i's such that the ratio of the expected cost of SR and the cost of VMIN for s_x is some constant $1 + \gamma$, i.e.,

$$E[C_{SR}(s_x)] = \begin{cases} (1+\gamma)xU & \text{if } 1 \le x \le \lfloor R/U \rfloor \\ (1+\gamma)R & \text{if } x > \lfloor R/U \rfloor. \end{cases}$$

Let $e_h = (1-1/h)^{-h}$. As $h \to \infty$, $e_h \to e$, which is $2.718\ldots$. As before, let $\alpha = \lfloor \frac{R}{U} \rfloor \frac{U}{R} \le 1$.

Lemma 1 *The constant γ satisfying the equations is $\frac{\alpha}{(e_{R/U})^\alpha - \alpha}$.*

Theorem 5 *The competitive ratio of the randomized algorithm SR is $\frac{(e_{R/U})^\alpha}{(e_{R/U})^\alpha - \alpha}$, which approaches $\frac{e^\alpha}{e^\alpha - \alpha} \le \frac{e}{e-1} \approx 1.58$ as R/U increases.*

Lower Bound

Algorithm SR is strongly competitive, i.e., its ratio matches the lower bound of competitive ratios of any randomized algorithms.

Theorem 6 *For any online randomized algorithm B, there are arbitrarily long strings σ such that*

$$E[C_B(\sigma)] \ge \frac{(e_{R/U})^\alpha}{(e_{R/U})^\alpha - \alpha} C_{VMIN}(\sigma).$$

Proof. We prove the lower bound by showing that there exist a string σ for which $E[C_B(\sigma)] \ge E[C_{SR}(\sigma)]$ for any online randomized algorithm B. Suppose that algorithm B keeps a referenced page for i time units with probability b_i.

The string σ that will be constructed consists of one page p. Suppose that page p is referenced at time t. We determine the next reference time as follows.

- If $\sum_{0 \le i \le x-1} b_i \ge \sum_{0 \le i \le x-1} r_i$ for some $x \le \lfloor R/U \rfloor + 1$, let z be the smallest such x. The next reference time will be $t + z$.
- If $\sum_{0 \le i \le x-1} b_i < \sum_{0 \le i \le x-1} r_i$ for all $x \le \lfloor R/U \rfloor + 1$, the next reference time z is $2\lfloor R/U \rfloor + 1$.

□

Conclusion

Fixed-space

- LRU and FIFO are strongly competitive (ratio k)
- Strongly competitive randomized algorithm (ratio H_k)

Variable-space

- WS is strongly competitive (ratio < 2)
- Strongly competitive randomized algorithm SR (ratio ≈ 1.58)

Further research

- Multi-level memory

The K-Server Problem

- Fiat, Rabani and Ravid (1990): competitive ratio $O(k^k)$
- Grove (1991): $O(k2^k)$
- The work function algorithm (WFA) that had long been conjectured to be strongly competitive [CL92]
- Koutsoupias and Papadimitriou (1994): proved that WFA is $(2k-1)$-competitive

M. Chrobak and L.L. Larmore, An optimal online algorithm for k-servers on trees, SIAM J. Comput. 20, 1 (1991), 144-148.

Financial Problems

Ran El-Yaniv, Competitive solutions for online financial problems, ACM Computing Surveys 30, 1 (1998), 28-69.

- Online search and one-way trading
  - Preservation price policy
  - Expo
  - Threat-based policy
- Portfolio selection
  - Buy-and-hold vs. Market timing
  - Constant rebalancing
- Risk management
  - One-way trading with risk management