A Short Proof of Optimality for the **MIN** Cache Replacement Algorithm

Benjamin Van Roy
Stanford University

November 10, 2006

Abstract

The **MIN** algorithm is an offline strategy for deciding which item to replace when writing a new item to a cache. Its optimality was first established by Mattson, Gecsei, Slutz, and Traiger [2] through a lengthy analysis. We provide a short and elementary proof based on a dynamic programming argument.

Keywords: analysis of algorithms, on-line algorithms, caching, paging

1 The MIN Algorithm

Consider the management of a cache over T time periods given advance knowledge of requests $\omega_0, \omega_1, \ldots, \omega_{T-1}$ from a set Ω of items stored in slower memory. Let $S_t \subset \Omega$ denote the set of items stored in the cache just before ω_t is requested. If $\omega_t \in S_t$ then $S_{t+1} = S_t$. Otherwise, a decision must be made to evict one item from the cache, which is replaced by ω_t. The objective is to maximize the number of hits: $\sum_{t=0}^{T-1} \mathbf{1}(\omega_t \in S_t)$.

The **MIN** algorithm chooses to evict an item in the cache whose next request occurs furthest in the future. If there are multiple items that will never again be requested one of them is chosen arbitrarily. Mattson, Gecsei, Slutz, and Traiger [2] establish that this algorithm is optimal.[1] However, their proof is somewhat long and complicated. The textbook *Randomized Algorithms* [3] offers an excellent account of several cache replacement algorithms and their analyses. In the case of the **MIN** algorithm, the authors cite the optimality result of [2] without providing the proof, which they mention to be "nontrivial." Here we offer a short and elementary proof based on a dynamic programming argument.

---

[1] In [2], this algorithm is referred to as the **OPT** algorithm. In that work, the term **MIN** identifies another algorithm originally proposed in [1].

2 Proof of Optimality

Let $J_t(S_t)$ denote the maximum possible number of hits to occur starting with the tth request. The dynamic programming recursion takes the form

$$J_t(S_t) = \mathbf{1}(\omega_t \in S_t) + \max_{S_{t+1} \in U_t(S_t)} J_{t+1}(S_{t+1}), \qquad \text{for } t < T, S_t \subset \Omega,$$

with the boundary condition $J_T(S_T) = 0$, where

$$U_t(S_t) = \{S \subseteq S_t \cup \omega_t : \omega_t \in S, |S| = |S_t|\},$$

is the set of possible successive states. The state $S_{t+1} \in U_t(S_t)$ is an optimal choice if and only if it attains the maximum in the above equation.

Denote the time of an item's next request by $\tau_t(\omega) = \min\{t \leq z < T : \omega_z = \omega\}$. (The minimum of an empty set is taken to be ∞.) Consider two cache states $S, S' \subset \Omega$ with $|S| = |S'|$. A matching is a bijection $h : S \mapsto S'$. Let

$$d_t(S, S') = \min_{h \in \text{matchings}} \Big|\{\omega \in S | \tau_t(\omega) > \tau_t(h(\omega))\}\Big|.$$

This is the minimum among matchings of the number of items in S requested after matched items in S'.

The **MIN** algorithm evicts the item to be requested furthest in the future. Hence, if S_{t+1} is chosen by the **MIN** algorithm and $Z \in U_t(S_t)$ is some other feasible choice then $d_{t+1}(S_{t+1}, S') \leq d_{t+1}(Z, S')$ for any cache state S'. In other words, $S_{t+1} \in \operatorname{argmin}_{Z \in U_t(S_t)} d_{t+1}(Z, S')$ for all S'.

The following lemma shows how d_t can be used to bound differences among values of cache states.

Lemma 1. *$J_t(S') - J_t(S) \leq d_t(S, S')$ for all $t \leq T$ and $S, S' \subset \Omega$ with $|S| = |S'|$.*

Proof: Since $J_T(S) = J_T(S') = 0$ and $d_T(S, S') = 0$, the result holds for $t = T$. We proceed by weak induction. Fix $t < T$ and assume $J_{t+1}(S') - J_{t+1}(S) \leq d_{t+1}(S, S')$ for all $S, S' \subset \Omega$ with $|S| = |S'|$.

Let $S_t = S$ and $S'_t = S'$. Let $S'_{t+1} \in U_t(S'_t)$ be chosen by an optimal strategy. Let $S_{t+1} \in U_t(S_t)$ be chosen by the **MIN** algorithm, and note that this implies $S_{t+1} \in \operatorname{argmin}_{Z \in U_t(S_t)} d_{t+1}(Z, S'_{t+1})$. We study the relationship between $d_t(S_t, S'_t)$ and $d_{t+1}(S_{t+1}, S'_{t+1})$ in four cases:

Case 1: Both S_t and S'_t are hit by ω_t. Neither cache state changes, so $d_{t+1}(S_{t+1}, S'_{t+1}) = d_t(S_t, S'_t)$.

Case 2: Neither S_t nor S'_t are hit by ω_t. If S'_t evicts an item, S_t can evict the same item. It follows that $d_{t+1}(S_{t+1}, S'_{t+1}) \leq d_t(S_t, S'_t)$.

Case 3: Only S_t is hit by ω_t. S'_t evicts an item and at best this improves its relative standing by 1; that is, $d_{t+1}(S_{t+1}, S'_{t+1}) \leq d_t(S_t, S'_t) + 1$.

Case 4: Only S'_t is hit by ω_t. S_t was previously disadvantaged relative to S'_t by not holding ω_t but now by replacing the item to be requested furthest in the future with ω_t, this disadvantage vanishes. Hence, $d_{t+1}(S_{t+1}, S'_{t+1}) \leq d_t(S_t, S'_t) - 1$.

These four relations together imply that

$$d_{t+1}(S_{t+1}, S'_{t+1}) \leq d_t(S_t, S'_t) + \mathbf{1}(\omega_t \in S_t) - \mathbf{1}(\omega_t \in S'_t).$$

Using this relation, the dynamic programming recursion, and our inductive hypothesis, we obtain

$$\begin{aligned}
J_t(S'_t) &= \mathbf{1}(\omega_t \in S'_t) + J_{t+1}(S'_{t+1}) \\
&\leq \mathbf{1}(\omega_t \in S'_t) + d_{t+1}(S_{t+1}, S'_{t+1}) + J_{t+1}(S_{t+1}) \\
&\leq \mathbf{1}(\omega_t \in S_t) + d_t(S_t, S'_t) + J_{t+1}(S_{t+1}) \\
&\leq \mathbf{1}(\omega_t \in S_t) + \max_{Z \in U_t(S_t)} J_{t+1}(Z) + d_t(S_t, S'_t) \\
&= J_t(S_t) + d_t(S_t, S'_t).
\end{aligned}$$

$\square$

Given S_t, let S_{t+1} be a successive cache state selected by the **MIN** algorithm and let S'_{t+1} be a successive cache state selected by an optimal strategy. Since S_{t+1} minimizes $d_{t+1}(S_{t+1}, S'_{t+1})$ over the set $U_t(S_t)$ which contains S'_{t+1}, $d_{t+1}(S_{t+1}, S'_{t+1}) = 0$. By the optimality of S'_{t+1} and Lemma 1,

$$0 \leq J_{t+1}(S'_{t+1}) - J_{t+1}(S_{t+1}) \leq d_{t+1}(S_{t+1}, S'_{t+1}).$$

It follows that $J_{t+1}(S'_{t+1}) = J_{t+1}(S_{t+1})$, and therefore, a decision made by the **MIN** algorithm is optimal.

Acknowledgments

Balaji Prabhakar introduced the author to the **MIN** algorithm, its history, and the interest in obtaining a simpler proof of optimality.

References

[1] L. A. Belady. A study of replacement algorithms for virtual storage computers. *IBM Systems Journal*, 5(2):78–101, 1966.

[2] R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger. Evaluation techniques for storage hierarchies. *IBM Systems Journal*, 9(2):78–117, 1970.

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