Caching Algorithms and Rational Models of Memory

Avi Press (avipress@berkeley.edu)
Michael Pacer (mpacer@berkeley.edu)
Thomas L. Griffiths (tom_griffiths@berkeley.edu)
Department of Psychology, University of California, Berkeley, Berkeley, CA 94720 USA

Brian Christian (brian.christian@berkeley.edu)
Institute of Cognitive and Brain Sciences, University of California, Berkeley, Berkeley, CA 94720 USA

Abstract

People face a problem similar to that faced by algorithms that manage the memory of computers: trying to organize information to maximize the chance it will be available when needed in the future. In computer science, this problem is known as "caching". Inspired by this analogy, we compared the properties of a model of human memory proposed by Anderson and Schooler (1991) and caching algorithms used in computer science. We tested each algorithm on a dataset relevant to human cognition: headlines from the *New York Times*. In addition to overall performance, we investigated whether the algorithms from computer science replicated the well-documented effects of recency, practice, and spacing on human memory. Anderson and Schooler's model performed comparably to the worst caching algorithms, but was the only model that captured the spacing effects seen in human memory data. All models showed similar effects of recency and practice.

Keywords: memory, caching algorithms, rational analysis

Introduction

Between our own experience forgetting things and the volumes of literature describing how fallible our memory is, it is easy to be critical of human memory. On the other hand, those with hyperthymestic syndrome (who can't help but remember excessive details about every day of their lives) struggle to deal with their inability to forget useless information (Parker, Cahill, & McGaugh, 2006). How does our brain know what information should be kept and what should be forgotten?

An analogue of the problem faced by human memory – as pointed out by Anderson and Milson (1989) – is a library trying to determine which books to keep in its collection. With finite shelf space, the library needs to decide which books are most likely to be needed in the future, relegating the others to long-term storage. Anderson and Milson used this observation as inspiration for a *rational* model of human memory, which prioritizes stored information by how likely it is to be needed in the future. Anderson and Milson showed that this approach captured several phenomena of human memory.

However, the rational model proposed by Anderson and Milson deviates from the library analogy in allowing infinitely many items to be stored in memory, with retrieval failure being the result of the need probability of a target item being below a threshold determined by the cost of searching. But an alternative construal of the problem is much closer to the original analogy: What if human memory really did have finite capacity? How *should* we choose what to forget?

The problem of choosing what to forget is an instance of what computer scientists term *caching*. A cache is a small yet fast block of memory (normally due to its hardware design and proximity to the processor), where a limited set of items can be stored. Every time the computer needs data that is not in the cache it must fetch it from somewhere that will take far more time to access, such as a hard disk. Whenever an item is added to the cache, the computer must use an algorithm to decide which other item to evict. The computer is thus constantly deciding what to forget, managing its limited memory resources to maximize the probability of the cache containing the items most likely to be needed.

In this paper, we explore the consequences of a rational analysis of human memory that assumes finite, rather than infinite, capacity. By looking at what happens when an environment is filtered through a finite cache, we can determine whether the statistical patterns corresponding to practice, recency, and spacing effects are relevant to successfully managing a finite memory. This is also potentially valuable for computer science, as we can see whether a caching scheme based on human memory improves on existing algorithms.

The plan of the paper is as follows. First, we summarize the memory phenomena that have been used to evaluate rational models, and describe the models themselves. Next, we define the problem of caching and introduce a set of caching algorithms. These algorithms are then evaluated in a set of four simulations. The first assesses overall performance. The others explore practice, recency, and spacing effects in turn.

Human Memory

Following Anderson and Schooler (1991), we will focus on three properties of human memory: practice, recency, and spacing. We review these phenomena, then turn to how they have been explained using rational models of memory.

Behavioral Phenomena

Practice The first two phenomena – practice and recency – are based on data collected by Ebbinghaus (1885/1913). Through rigorous self-experimentation, Ebbinghaus was able to discover some of the most basic aspects of human memory. The practice effect is simply that more times an item has been encountered, the more likely it can be recalled. Subsequent work has attempted to identify the form of the relationship between practice and retention, and found that a power-law best captures this relationship (Newell & Rosenbloom, 1981).

Recency Ebbinghaus also noted that the more recently an item was encountered, the more likely it can be recalled.

The form of this relationship has been debated (e.g., Loftus, 1985), but Anderson and Schooler (1991) showed that the data from Ebbinghaus (1885/1913) followed a power law.

Spacing Memory research has taken significant steps since the original work conducted by Ebbinghaus, with one important discovery being the existence of spacing effects (e.g., Glenberg, 1976). The number and recency of encounters with items are not sufficient to determine performance: people are also sensitive to the amount of time that passes between successive encounters. This pattern can be described by the amount of time passing in between individual encounters (study lag) and how much time passed since the last encounter (test lag). For a short test lag, recall is better with a smaller study lag. However, for longer test lag, recall actually increases with study lag. A longer interval between encounters thus seems to establish longer-lasting memories.

Rational Models of Memory

Anderson and Milson (1989) proposed that the problem human memory faces is organizing information in order to facilitate retrieval. More formally, if accessing an item in memory incurs a cost C and finding the target item results in gain G, then a rational agent should access items in decreasing order of the probability p that they are the target item, stopping when $pG < C$. The challenge, then, is to calculate the probability that an item is likely to be the target. One component of this is the probability that an item is likely to be needed at a given moment – the *need probability*.

Subsequent work by Anderson and Schooler (1991) suggested that human memory need not perform complex calculations of need probabilities. They showed that effects of practice, recency, and spacing are consistent with statistical patterns that appear in human environments – such as which words appear in the headlines of articles in the *New York Times*. They calculated the probability that a word would appear in a headline as a function of its pattern of occurrences in headlines over the previous 100 days. This analysis showed that need probability increased as a power-law in the number of previous occurrences (a practice effect), decreased as a power-law in the amount of time since the last occurrence (a recency effect), and was higher for less recent items when those items had been more widely spaced (a spacing effect).

On the basis of these results, Anderson and Schooler proposed that a reasonable proxy for need probability could be defined by assigning a "strength" to each item in memory. The strength function they suggested was

$$S_{AS91} = A \sum_{i=1}^{n} s(t_i), \tag{1}$$

where A is a constant, n is the number of times the item has occurred, t_i is the time of the ith occurrence, $s(t_i) = t_i^{-d_i}$ and $d_i = \max[d_1, b(t_i - t_{i-1})^{-d_1}]$. being the how the strength from the ith occurrence decays with time. d_1 is a tunable parameter of the model. With $d_1 = 0.125$, the model was able to reproduce the practice, recency, and spacing effects, so we use this parameter setting in the analyses we present in this paper.

An Alternative Rational Analysis

The strength function proposed by Anderson and Schooler (1991) – like the need probability considered by Anderson and Milson (1989) – is simply used to prioritize items in memory. The set of items is assumed to be infinite, with search through memory proceeding in decreasing order of strength and terminating when strength falls below a threshold. While the set of things a person can remember is not explicitly limited, the recency effect imposes an implicit upper bound. As time passes, strength of memories are decaying, and eventually their strengths fall below the threshold.

An alternative rational analysis might consider a different cost function: what if memory is finite, and a cost is incurred for failing to retrieve an item? The goal then is to maximize the chance that an item is already contained in memory when it is needed. Under this alternative view, need probabilities remain critical – memory should contain only those items with the highest need probabilities. But forgetting is also obligatory, as new items force out old.

While the change from an infinite capacity with a cost for searching to a finite capacity with a cost for failure might seem minor, it potentially has significant effects on the resulting models. For example, even though Anderson and Schooler showed that practice, recency, and spacing all affect need probability, their degree of influence could differ. Optimally managing finite memory resources might require attending to some of these factors more than others.

In the remainder of the paper, we explore the consequences of adopting this alternative view of the problem faced by human memory. Importantly, adopting this view allows us to explore potential links between human memory and the algorithms used for cache management by computers.

Caching Algorithms

Many caching algorithms can be cast in similar terms to Anderson and Schooler's (1991) memory model, assigning a strength to items and evicting the item with the lowest strength when a new item is added to the cache. This provides a natural basis for comparison of these approaches. In this section, we summarize a set of algorithms that computer scientists have developed to solve the problem of memory caching. Some of the cache policies encode recency, others frequency, and some try to balance both. In the remainder of the paper, we evaluate the performance of these algorithms on data from the human cognitive environment (ie., the *New York Times*) and compare them to Anderson and Schooler's (1991) model (Equation 1; henceforth AS91) with respect to the way in which recency, practice, and spacing affect recall.

Least Recently Used (LRU)

Perhaps the simplest of caching algorithms, LRU evicts the item that was used least recently (Belady, 1966). Because the only information needed to implement LRU is order of uses,

it can be implemented with just a simple list that preserves this information. A strength function consistent with LRU is

$$S_{LRU} = \frac{1}{t_{current} - t_n}. \quad (2)$$

where $t_{current}$ is the current time and t_n is the time of the nth (ie., last) occurrence of the item. LRU has proved to be very successful in computer science applications since it can exploit the "locality" of computer behavior, where an item being used once makes it likely to be used again in a short interval.

Least Frequently Used (LFU)

Another straightforward (yet extreme) approach is to evict the item that has been used least. LFU's strength function is

$$S_{LFU} = n, \quad (3)$$

where n is the number of times the item has occurred. An item's strength thus grows the more times it is used. LFU can be very effective when items are used often but not necessarily in temporal proximity. A major drawback to LFU is that the cache can become littered with items that were once extremely popular but might never be used again. This issue can actually be quite extreme; a pure LFU system is fairly uncommon in computer science.

LRU-2

LRU-2 evicts the item with the least recent penultimate use. Both LRU and LRU-2 can be described as being part of the LRU-k family of algorithms (O'Neil, O'Neil, & Weikum, 1999), where the item with the least recent kth use is evicted (LRU being LRU-k for $k = 1$). This corresponds to

$$S_{LRU-k} = \frac{1}{t_{current} - t_{n-k+1}} \quad (4)$$

where t_i is the time of the ith occurrence. Simple LRU does not account for frequency, so LRU-k is a way to introduce frequency into an algorithm that is also sensitive to recency. Behavior becomes closer to LFU as k increases, and closer to LRU as k approaches 1. A common compromise between frequency and recency is to take $k = 2$ (O'Neil et al., 1999).

2Q

2Q (Johnson & Shasha, 1994) tries to find a balance between accounting for recency and frequency by splitting the cache into two queues (and technically a third queue, but that will be mentioned later). The first queue is managed as an LRU queue. If a hit (ie., successful retrieval of an item) occurs in this queue, the item is promoted to the second queue, which is managed as a LFU queue. The LFU queue has a predefined maximum size, so items will be evicted from the LFU queue if the queue is larger than the predefined size, and evicted from the LRU queue otherwise. This policy can be interpreted in an interesting way: The first queue could be thought of as short term memory, and items used enough get promoted to the second queue, long term memory (Atkinson & Shiffrin, 1968). If an item is in the queue that will be evicted from, then its strength will be equivalent to the LRU or LFU functions (depending on which queue it is). If the item is not in this queue, its strength is essentially infinite; it will never be evicted until the size of the LFU queue changes. An issue with 2Q is that it may be hard to estimate the appropriate size for the LFU queue ahead of time.

Adaptive Replacement Cache (ARC)

ARC (Megiddo & Modha, 2004) is very similar to 2Q but with the capacity to adapt to the environment. In ARC, the maximum size of the LFU queue changes as the data comes in. The algorithm keeps track of items that have been evicted from each queue. Upon a cache miss (ie., a failed retrieval) for an item that was recently evicted from one of the queues, ARC will make that queue larger, as it got rid of an item that it should have kept.

LRFU

LRFU (Kim, 2001) subsumes both LRU and LFU. Each item has a combined recency-frequency count which is its strength. Intuitively, an item's strength continually climbs each time it is used but that strength decays with time. The exact function can vary, but as suggested by Kim (2001) we used the strength function

$$S_{LRFU} = \sum_{i=1}^{n} \frac{1}{2}^{\lambda(t_{current} - t_i)}, \quad (5)$$

where λ is a tunable parameter. For our purposes, $\lambda = 0.001$ worked well. We note that LRFU particularly closely resembles the memory model proposed by Anderson and Schooler (1991), with an exponential rather than a power-law decrease in strength (compare Equations 1 and 5).

Random

A simple alternative to these complex caching algorithms is to assign each item a random strength. This means that a random item will be evicted whenver a new item is introduced. This provides a reasonable lower bound on performance.

Belady's Algorithm

Belady's algorithm (Belady, 1966) is the optimal caching algorithm, providing an upper bound on performance. However, it achieves this optimality by being able to see the future, providing it with an unfair advantage over other algorithms. The algorithm works simply by evicting the item that will be used the furthest in the future. This is not a real caching policy, because we will never know the sequence of accesses ahead of time. We use it in this paper in order to compare each algorithm to the optimal caching policy, and to examine what environmental statistics it exploits.

Simulation 1: Overall Miss Rate

Our first analysis compared the overall performance of all of the algorithms as caching policies, focusing on the rate of cache misses on data from a human environment.

Methods

A dataset of headlines from the *New York Times* from January 1986 to December 1987 was used to test the various caching algorithms. This is one of the datasets used by Anderson and Schooler (1991) in their analysis of environmental statistics. Words from the headlines were sequentially cached, each word being treated as a separate item. Words were sanitized of punctuation, and made lower-case. We calculated the ratio of misses to hits for a range of cache sizes. We also ran a test where we removed the top ten most used English words ("the", "and", etc.) according to Wikipedia. The differences were negligible, and thus not included.

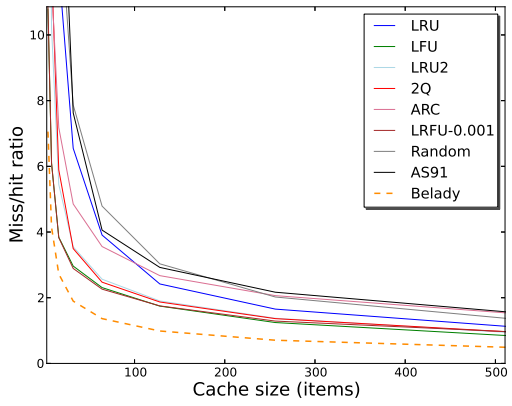

Figure 1: Overall miss rates on words in headlines from the *New York Times* as a function of cache size. Miss/hit ratio is the ratio of the number of misses to the number of hits.

Results and Discussion

The miss to hit ratio dropped at a similar rate across caching policies, as shown in Figure 1. Anderson and Schooler's (1991) model performed similarly to the worst algorithms, being only slightly better than random eviction. LFU and LRFU were the best algorithms for this data set, producing performance closest to Belady's algorithm. LFU performed worse than LRFU for small cache sizes, but then overtook it as cache size approached and exceeded 256 items.

The high performance of LFU is surprising, given that LRU generally outperforms LFU in comparisons based on calls to computer memory (e.g., Kim, 2001; Megiddo & Modha, 2004). The superior performance of LFU on this dataset may reflect two important things about human language. First, word frequency follows a power-law distribution, with a few words accounting for a large proportion of overall occurrences (Zipf, 1949). Second, this distribution is only be subject to weak influences of locality. For example, "the" will remain a high frequency word even if "Gaddafi" appears in the headlines over a series of days.

Simulation 2: Practice

Our second analysis examined whether the practice effect – higher probabilities for items that appear more frequently – was maintained when data were filtered through a cache.

Methods

The *New York Times* headlines were used again for this simulation (and all subsequent simulations). We used a cache of a fixed size, $|C| = 128$ items, although effects were similar for other cache sizes. For each 100 day interval in the dataset, we looked at a word's probability of being in the cache on the 101st day, based on the number of times it was used in the interval. This is very similar to what Anderson and Schooler's (1991) analysis, except that they looked at what words were actually used on the 101st day.

Results and Discussion

An effect of practice was observed for all cache policies, as shown in Figure 2. Effects of practice were, as might be expected, strongest for LFU and weakest for LRU. LFU and then LRFU were most affected by practice, both even more so than Belady's algorithm. None of the algorithms showed a clear power-law relationship between practice and hit probability, but this may be due to the rapid saturation of the performance curves.

Simulation 3: Recency

We next examined whether a recency effect – higher probabilities for more recent items – was shown by the caching algorithms.

Methods

The same methods were used in this simulation as used in Simulation 2. This time we calculated probability that a word was in the cache on the 101st day given the word's recency (ie., how many days had passed since it had been used).

Results and Discussion

All algorithms showed a strong recency effect, as shown in Figure 3, although LFU and AS91 were the only algorithms to show a power-law relationship. Not surprisingly, the drop-off was steepest for LRU. Interestingly, Belady's algorithm showed a recency effect that did not follow a power-law relationship, indicating that a faster decrease in the influence of recency is optimal for this dataset.

Simulation 4: Spacing

Our final simulation examined whether the caching algorithms were sensitive to spacing, analyzing the interaction between how recently a word had been encountered and how much time passed between successive instances of that word.

Methods

Our analysis followed the approach taken by Anderson and Schooler (1991). We identified words that had appeared exactly twice in the previous 100 days, and divided them into those that had a "short lag" (10 days or less between occurrences) and those that had a "long lag" (more than 10 days between occurrences). We then examined the effect of recency for these two sets of words.

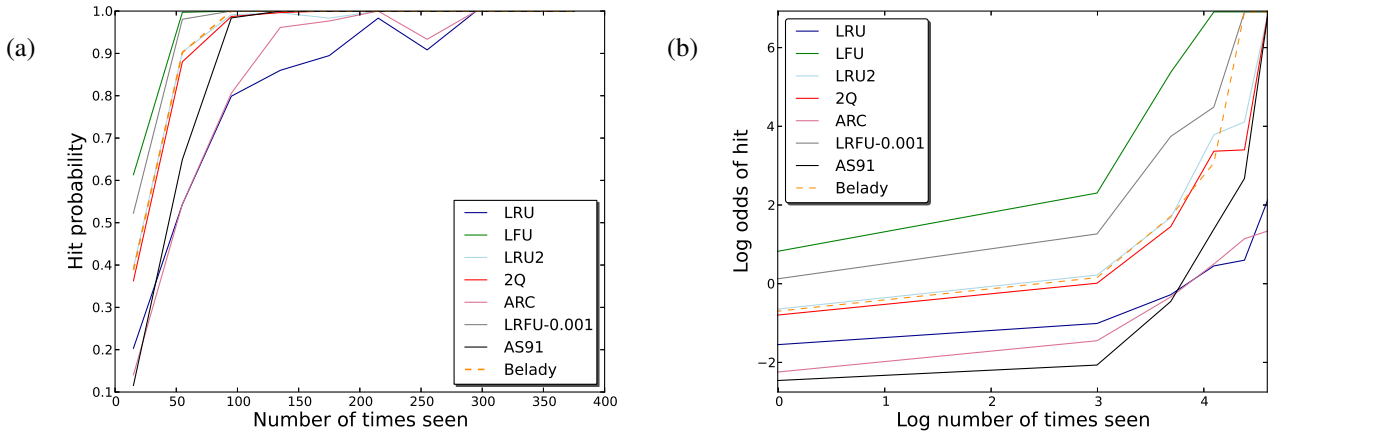

Figure 2: Practice effects. (a) Probability of a cache hit as a function of number of occurrences in the preceding 100 days. (b) Log odds of a hit as a function of log number of occurrences.

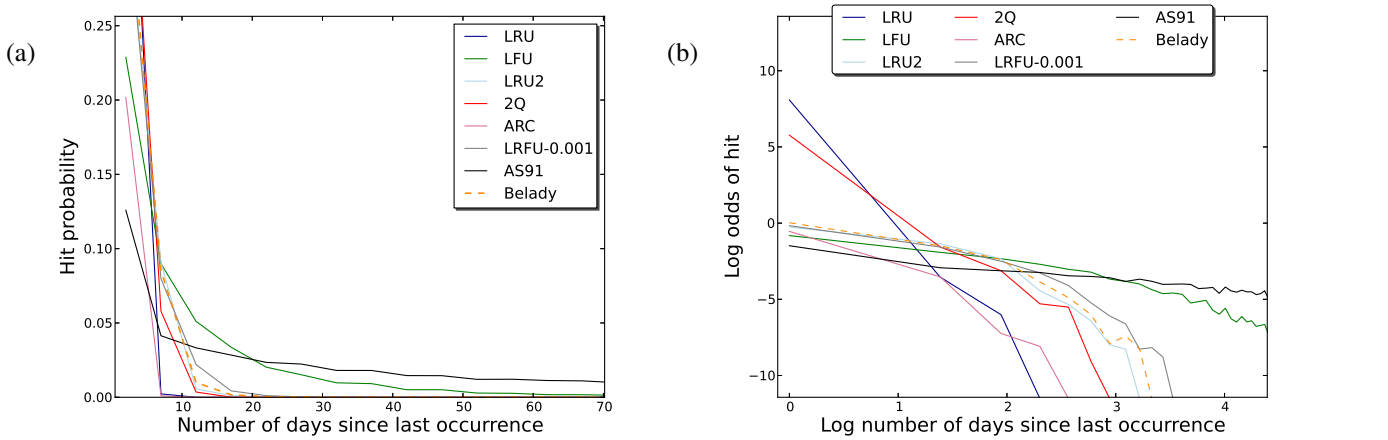

Figure 3: Recency effects. (a) Probability of a cache hit as a function of number of days since last occurrence. (b) Log odds of a hit as a function of log of number of days since last occurrence.

Results and Discussion

The results are shown in Figure 4. In short, no caching algorithms other than AS91 showed a spacing effect. Only AS91 showed a higher hit rate for less recent items with a long lag.

Conclusions

Our analysis of Anderson and Schooler's (1991) model as a caching policy, and of algorithms from computer science as potential models of human memory, yields three conclusions. First, Anderson and Schooler's model performed poorly as a caching policy on the *New York Times* dataset. This is not a critique of the model, as it was not designed to solve this problem, but it is surprising that a model that was designed to capture need probabilities performs so poorly. In particular, it shows that there is potentially a bigger gap than might have been anticipated between the construals of memory as a problem of caching as opposed to prioritization.

Second, the high performance of LFU and related algorithms is at odds with previous results in computer science, and suggests that different caching algorithms may be ideal in human environments than computer environments. This is an interesting finding that deserves further investigation through the analysis of other datasets derived from human environments. However, it creates an opportunity to explore other algorithms that may be effective for caching in contexts relevant to humans.

Finally, while analogues of practice and recency effects appear in the behavior of caching algorithms, tracking the spacing of items does not appear to improve performance for this dataset in particular. Furthermore, higher-performing caching algorithms did not show a power-law effect of recency. The inclusion of Belady's algorithm in our analysis

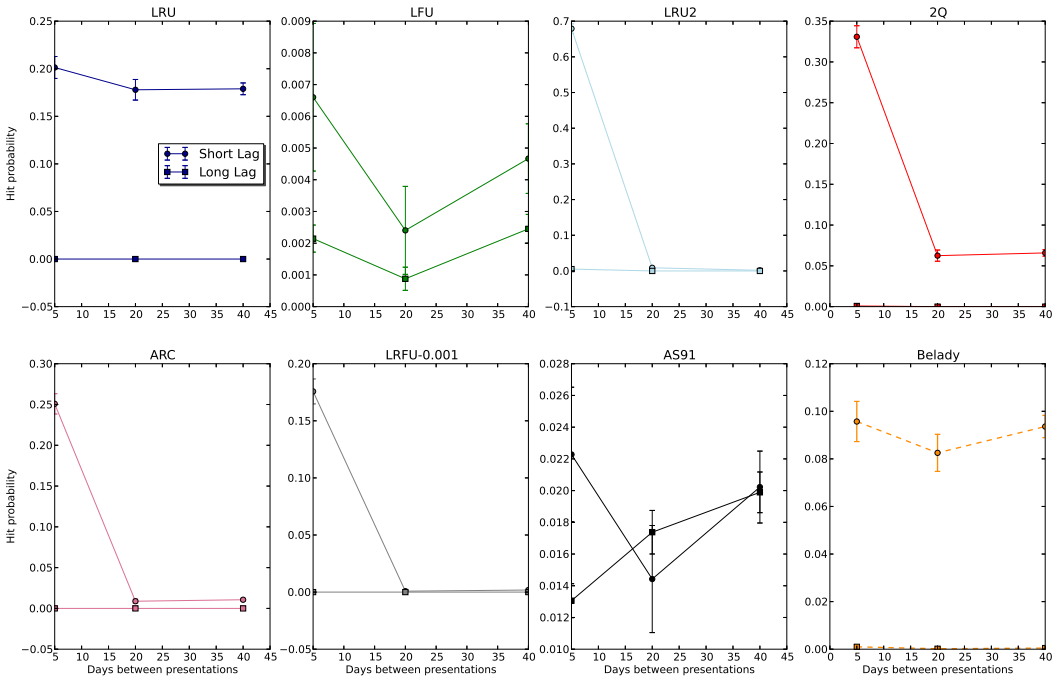

Figure 4: Spacing effects. Probability of a cache hit as a function of number of days since last occurrence, plotted separately for pairs of occurrences separated by a short lag (up to 10 days) and a long lag (more than 10 days) with 1 standard error.

is particularly instructive, as it indicates what statistical patterns are relevant to *optimal* performance, and shows neither spacing nor a power-law for recency. To the extent that these phenomena appear in human memory, and human memory is assumed to be adaptive, these findings provide evidence that the assumption of infinite capacity may be more appropriate as the basis for a rational analysis.

Acknowledgments. This work was supported by grant number SMA-1228541 from the National Science Foundation.

References

Anderson, J. R., & Milson, R. (1989). Human memory: An adaptive perspective. *Psychological Review*, *96*(4), 703.

Anderson, J. R., & Schooler, L. J. (1991). Reflections of the environment in memory. *Psychological science*, *2*(6), 396–408.

Atkinson, R. C., & Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes. *The psychology of learning and motivation*, *2*, 89–195.

Belady, L. A. (1966). A study of replacement algorithms for a virtual-storage computer. *IBM Systems Journal*, *5*(2), 78–101.

Denning, P. J. (1980). Working sets past and present. *IEEE Transactions on Software Engineering*, *6*(1), 64–84.

Ebbinghaus, H. (1885/1913). *Memory: A contribution to experimental psychology* (No. 3). Teachers college, Columbia university.

Glenberg, A. M. (1976). Monotonic and nonmonotonic lag effects in paired-associate and recognition memory paradigms. *Journal of Verbal Learning and Verbal Behavior*, *15*(1), 1-16.

Johnson, T., & Shasha, D. (1994). 2Q: A low overhead high performance buffer management replacement algorithm. *VLDB '94 Proceedings of the 20th International Conference on Very Large Data Bases*, 439-450.

Kim, C. S. (2001). LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies. *IEEE transactions on Computers*, *50*(12), 1352–1361.

Loftus, G. R. (1985). Evaluating forgetting curves. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *11*(2), 397-406.

Megiddo, N., & Modha, D. S. (2004). Outperforming LRU with an adaptive replacement cache algorithm. *Computer*, *37*(4), 58–65.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. *Cognitive skills and their acquisition*, 1–55.

O'Neil, E. J., O'Neil, P. E., & Weikum, G. (1999). An optimality proof of the LRU-K page replacement algorithm. *Journal of the ACM (JACM)*, *46*(1), 92–112.

Parker, E. S., Cahill, L., & McGaugh, J. L. (2006). A case of unusual autobiographical remembering. *Neurocase*, *12*, 35-49.

Zipf, G. K. (1949). *Human behavior and the principle of least effort.* New York: Addison-Wesley.