Cost-Aware WWW Proxy Caching Algorithms

Greedy Dual-Size Algorithm:

Variable On: Greedy-Dual Algorithm (uniform cost variable cost cache replacement)
Incorporates: locality with cost and size concerns
Outperforms: existing web cache replacement algorithms (with appropriate cost definition).
Focus Areas: hit ratios, latency reduction and network cost reduction.
Improve: Performance of main-memory caching of Web documents

Definitions Used:
k : maximum number of bytes that can fit in the cache.
Locality: refers to temporal locality i.e. the time when a document is referenced compared to current time.

Terminologies Used:
Online Algorithm/Offline Algorithm/Competitive Analysis: (source: http://en.wikipedia.org/wiki/Online_algorithm)
In computer science, an online algorithm is one that can process its input piece-by-piece in a serial fashion, i.e., in the order that the input is fed to the algorithm, without having the entire input available from the start. In contrast, an offline algorithm is given the whole problem data from the beginning and is required to output an answer which solves the problem at hand. (For example, selection sort requires that the entire list be given before it can sort it, while insertion sort doesn't.)

Because it does not know the whole input, an online algorithm is forced to make decisions that may later turn out not to be optimal, and the study of online algorithms has focused on the quality of decision-making that is possible in this setting. Competitive analysis formalizes this idea by comparing the relative performance of an online and offline algorithm for the same problem instance

Introduction:
Assumptions: majority of Web documents requested are static documents

Where is caching occurring?
- At various network points.
- At HTTP proxies (intermediaries between browser processes and web servers)

Benefits of Caching:
- reduces network traffic (hit ratio for Web proxy caches can be as high as over 50%)
- average latency of fetching Web documents
- load on busy Web servers

Difference between Web Caching & Conventional Paging Problems:
- Web caching is variable-size caching (web documents vary dramatically in size depending on the information they carry (text, image, video, etc.).
- Web pages take different amounts of time to download, even if they are of the same size (download latency).
- Access streams seen by the proxy cache are the union of web access streams from tens to thousands of users (instead of coming from a few programmed sources as in the case of virtual memory paging)

Example Improvements:
- replacement algorithm can favor documents travelling through the expensive links over cheap documents.
- caching algorithm can retain more documents which must travel on congested paths.
- preferentially caching documents that travel more hops. (NETWORK COST CONSIDERATION)

Algorithms Used Till Now: LRU (Least Recently Used): with TTL (Time to Live). (None other algo address the n/w cost)
Existing Results

- In the variable-size case, no such offline algorithm is known. In fact, it is known that determining the optimal performance is NP-hard (approximated to logarithmic factor).

- For cost consideration, GreedyDual is a range of algorithms which include a generalization of LRU and a generalization of FIFO. (Optimal according to competitive ratio).

- The studies offer no conclusion on which algorithm a proxy should use. (LRU, LFU, SIZE, LRU-threshold, Log (Size) + LRU, Hyper-G, Lowest Latency First, Hybrid, LRV).

Figure 1 shows: probability of a document being accessed again as a function of the time since the last access to this document.

Figure 2 shows: each user tends to re-access recently-read documents, and re-access documents that are read on a daily basis (note the spikes around 24 hours, 48 hours, etc. in fig.)

Probability of re-reference to documents referenced exactly t minutes ago can be modeled as k=t, where k is a constant.

Users' interests overlap in time: comparing figures 2 and 1, we can see that for the same t, the percentage in figure 1 is higher than that in figure 2. This indicates that part of the locality observed by the proxy comes from the fact that the proxy sees a merged stream of accesses from many independent users, who share a certain amount of common interests.

Conclusion:
In the absence of cost and size concerns, LRU is the optimal online algorithm for reference streams exhibiting good locality. However in web context other factors like:

- Replacing a more recently used but large file can yield a higher hit ratio than replacing a less recently used but small file.
- Similarly, replacing a more recently used but inexpensive file may yield a lower total cost than replacing a less recently used but expensive file.
Thus, we need an algorithm that combines locality, size and cost considerations in a simple, online way that does not require tuning parameters according to the particular traces, and yet maximizes the overall performance.

**GreedyDual-Size Algorithm**

**Greedy Dual Algorithm:**
*Concerned with:* when pages in a cache have the same size, but incur different costs to fetch from a secondary storage.

**Definitions:**
- \( H: \) Value (Initial Value – cost of bringing the page into the cache)
- \( p: \) Cached page

**Policy:**
- *Replacement:* the page with the lowest \( H \) value, \( \min_H \) is replaced and then all pages reduce their \( H \) values by \( \min_H \).
- *Accessed:* If a page is accessed, its \( H \) value is restored to the cost of bringing it into the cache.

**Rationale:**
Thus, the \( H \) values of recently accessed pages retain a larger portion of the original cost than those of pages that have not been accessed for a long time. By reducing the \( H \) values as time goes on and restoring them upon access, the algorithm integrates the locality and cost concerns in a seamless fashion.

**Greedy Dual-Size Algorithm:**
*Extension:* To incorporate the difference sizes of the document, we extend the GreedyDual algorithm by setting \( H \) to \( \frac{\text{cost}}{\text{size}} \) upon an access to a document.

**Definitions:**
- *Cost:* cost of bringing the document. Definition also depends on the goal of the replacement algorithm: cost is set to 1 if the goal is to maximize hit ratio, it is set to the downloading latency if the goal is to minimize average latency, and it is set to the network cost if the goal is to minimize the total cost.
- *Size:* size of document in bytes.

**Variations:**
- **Inflation Value:** GreedyDual-Size would require \( k \) subtractions when a replacement is made (where \( k \) is the number of documents in cache). Instead, keep ‘L’ – inflation value & let all future setting of \( H \) be offset by \( L \).

**Data Structure:**
Priority Queue on the documents, based on their \( H \) value.

**Time Complexity:**
- Handling a hit: \( O(\log k) \) - Search
- Handling an eviction: \( O(\log k) \) – the new document needs to be put at the right location in queue. (Search)
Proof of Online-Optimality:
The cost of cache misses under GreedyDual-Size is at most k times that under the offline optimal replacement algorithm, where k is the ratio of the cache size to the size of the smallest page.

**GreedyDual-Size is k-competitive:** \((\text{GreedyDual-Size}) \leq k \times \text{OPT(Offline Replacement Algorithm)}\)

Algorithm used to proof this is BALANCE which also solves the multi-cost uniform-size paging problem is k-competitive.

Performance Comparison

**Performance Metrics:**

a) **Hit Ratio**: number of requests that hit in the proxy cache as a percentage of total requests  
b) **Byte Hit Ratio**: no. of bytes that hit in the proxy cache as the percentage of the total number of bytes requested.  
c) **Latency Reduction**: percentage of the sum of downloading latency for the pages that hit in cache over the sum of all downloading latencies.  
d) **Hop Reduction**: ratio between the total no. of the hops of cache hits and the total no. of the hops of all accesses.  
e) **Weighted Hop Reduction**: corresponding ratio for the total number of hops times "packet savings" on cache hits.

A cache hit's packet saving is \(2 + \text{file_size}/536\), as an estimate of the actual number of network packets required if the request is a cache miss (1 packet for the request, 1 packet for the reply, and size/536 for extra data packets, assuming a 536-byte TCP segment size).

Hit Rate & Byte Hit Rate
The results show that clearly, GD-Size (1) achieves the best hit ratio among all algorithms across traces and cache sizes. It approaches the maximal achievable hit ratio very fast, being able to achieve over 95% of the maximal hit ratio when the cache size is only 5% of the total data set size.

Reduced Latency
GD-Size (1) is the best algorithm to reduce average latency. The high variance among loading latencies for the same document reduces the effectiveness of latency-conscious algorithms.

Network Costs
For hop reduction, GD-Size(hops) performs the best, and for weighted-hop reduction, GD-Size(weighted hops) performs the best.

This shows that GreedyDual-Size not only can combine cost concerns nicely with size and locality, but is also very flexible and can accommodate a variety of performance goals.

Summary
Based on the above results, we have the following recommendation. If the proxy wants high hit ratio or low average latency, GD-Size(1) is the appropriate algorithm. If the proxy desires high byte hit ratio as well, then GD-Size(packets) achieves a good balance among the different goals. If the documents have associated network or monetary costs that do not change over time, or change slowly over time, then GD-Size(hops) or GD-Size(weighted hops) is the appropriate algorithm. Finally, in the case of main memory caching of web documents, GD-Size(1) should be used because of its superior performance under small cache sizes.

Conclusion
This paper introduced a simple web cache replacement algorithm: GreedyDual-Size, and showed that it outperforms existing replacement algorithms in many performance aspects. It combines locality, cost and size considerations in a unified way without using any weighting function or parameter.