# Joint Broadcast Scheduling and User's Cache Management for Efficient Information Delivery


Chi-Jiun Su*
csu@hns.com
Hughes Network Systems
11717 Exploration Lane
Germantown MD 20876
Tel: (301) 601-7346
FAX: (301) 428-2750

Leandros Tassiulas
leandros@isr.umd.edu
Electrical Engineering Department
University of Maryland
College Park MD 20742
Tel: (301) 405-6620
FAX: (301) 314-9920



## Abstract

In information delivery through broadcasting, a server continuously pushes information in a broadcast channel and the users access it by tuning in and waiting until the information they are interested in is transmitted. The server follows a schedule that attempts to match the user access statistics in order to reduce the access latency. In case of inhomogeneous user populations with several different access profiles, the users have local cache to smooth out the mismatches of their profile with the broadcast schedule statistics.

In this work we propose a method for joint design of the server broadcast schedule and the user caching strategy such that the access delay is minimized. We identify a fluid model of the joint problem on which the joint optimization is performed and subsequently the dynamic schedule is designed to match the optimal fluid model parameters. It turns out that with joint design of the broadcast schedule and the user cache management policy significant performance improvement is achieved, particularly for inhomogeneous user populations.


## 1 Introduction

With recent advances in information technology, one-way push-based broadcast delivery is becoming a method of main interest for the distribution of information to a large user population. Applications that employ broadcast delivery are Boston Community Information System [8], Datacycle project at Bellcore [7], Hughes' DirectPC multimedia and package delivery system [10], PointCast's webcasting [14], Marimba's Castanet [13], AirMedia Live Internet Broadcast Network [3] and Intel's intercast [11] among others. Data to be disseminated in the applications includes news and weather information, traffic information, schedule information in airports and train stations, stock quotes and so on.

Under the broadcasting approach as shown in figure 1, a server continuously and repeatedly broadcasts data to a user community without any feedback about the user's needs due to the limited uplink communication capability from the user to the server. Data broadcast by the server are organized into units called *information items*. When a user needs a certain information item, it monitors the broadcast channel until the desired item is detected and captures it for use. There is some latency from the time the need of an information item arises until the time the item is actually broadcast by the server. This latency depends on the broadcast schedule of the server, as well as the user access pattern.

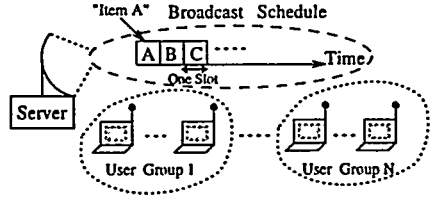


Figure 1: A Broadcast Data Delivery System in a Wireless Communication Environment

There has been a lot of work in the past on the problem of designing a broadcast schedule such that the average latency is minimized [5], [6], [18], [1], [9] and [15]. The approach is to determine the broadcast frequency of each information item in accordance with the user access frequency of the item and then to distribute the broadcast slots of each item as uniformly as possible. If there are more than one class of users with different access distributions of information items, then it is unavoidable that some classes will suffer large latency. An approach to reduce the latency to a desirable level for each user is to make use of local user storage.

If a user has local storage, it can retrieve information items from the broadcast and store them in its memory prior to the items being requested. If the user makes a request for one of the "prefetched" stored items, the response time for this request will be instantaneous. By selectively prefetching information items from the broadcast, the user is able to effectively minimize the mismatch between its access needs and server's broadcast schedule and the average latency of its information requests is reduced. Therefore, user's memory management becomes an important issue to consider in order to minimize the average response time of user's requests. As information items pass by in the broadcast, the

*The work of C.-J. Su was performed when the author was with University of Maryland, College Park.



MOBICOM 98 Dallas Texas USA

user has to decide whether an item will be prefetched and if it will, which item residing in the memory will be replaced with the newly prefetched item. The problem of user's memory management was considered in [16], [1], [2] and [4].

The following figure 2 shows the functionality of a server's broadcast scheduling policy and a user's cache management policy. The scheduling policy first transforms the set of user's access statistics into one aggregate access statistics and then employs it for scheduling of items in the broadcast. Given the access statistics of a user, the cache management policy decides if an item arriving in the broadcast will be prefetched. When a user needs an item, it checks if the item is in the cache. If not, it has to wait until the item appears in the broadcast.

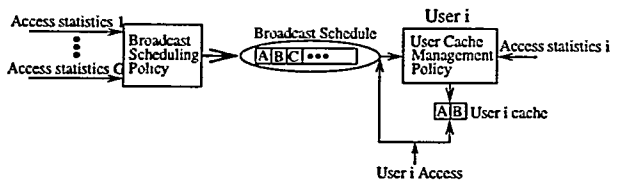


Figure 2: *The functionality of a broadcast scheduling policy and a user's cache management policy*

Due to the lack of uplink request channel, the user's access statistics have to be computed according to the user profiles generated by the users. Accruing profile information to estimate the user's access statistics is an interesting problem and merits further study and is beyond the scope of this paper.

The broadcast scheduling problem and the user's memory management problem are complementary to each other and they were addressed separately in literature. Traditional broadcast scheduling schemes does not take into account of user's caching although prefetching the broadcast items into user's cache can be effectively employed at the user's side in order to further reduce the mean response time of user's requests. In this paper, we consider an integrated approach which produces broadcast schedules and user's memory management schemes simultaneously based on the mix of data access pattern of user's groups. Under our joint approach, broadcast schedules are designed such that the schedules are conducive to user's caching and the overall mean response time can be considerably reduced when caching is done at user's side. A fluid model of the joint problem is identified on which the joint optimization is performed and subsequently the schedules are constructed according to the optimal fluid model parameters. Numerical results show that our joint approach can reduce the mean response time of the non-joint scheme up to 40%. We also show that the mean response time of user's requests under any broadcast scheduling and cache management policy is lowerbounded by the solution of the optimization problem. A lowerbound for the performance of any cache management strategy, when the server's broadcast schedule is known, is also provided which can be used as an benchmark to investigate the performance of a cache management policy. We present an example that shows that it is also necessary to take the group identity of an item (which group of users is interested in the item) into consideration for scheduling when prefetching is done at user's side. We also provide a two-level scheduling policy which is immune to pathological cases and performs slightly better than one-level scheduling in general.

The paper is organized as follows. The problem of joint broadcast scheduling and user's cache management is considered in section 2. An approach to jointly schedule broadcast and perform user's cache management is provided in section 3. Finally, in section 4, we investigate the performance of the joint approach by numerical experiments.

## 2 Joint Broadcast Scheduling and User's Memory Management

Time is slotted and one slot length is equal to the time to transmit one information item. Slot $n$ is the interval $[n, n+1)$. The sever broadcasts the information items according to a schedule $\{u_n\}_{n=0}^{\infty}$ where $u_n$ is the information item broadcast at slot $n$. Assume that there are $M$ possible information items. The index of the broadcast data can be transmitted to the users ahead of the actual broadcast data either through the same channel in which the data is transmitted or through a different one with lower data transmission rate. Therefore, in the following, we may assume that all the users know the whole broadcast schedule a priori.

A user is generating requests for information items according to its needs. When a request for some item $i$ is generated at some time $t$, then it is either satisfied immediately if the item resides in the local cache of the user or the user has to wait until the next time the item appears in the broadcast schedule. After the request is satisfied one way or the other, the user will generate another request for an information item after some random time. The latency from the time a request is generated until the item is transmitted by the server is the performance measure of interest in broadcast data delivery.

Let's assume that there are $G$ groups of users each of which is made up of users with an identical request generation process. If the user population of each group, $g$, $g = 1, ..., G$, is large enough, then we may assume that the request generation process of each group is stationary with constant rate $\lambda^g$ requests per slot. A request from group $g$ is for item $i$ with probability $b_i^g$, $i = 1, ..., M$ where $\sum_{i=1}^{M} b_i^g = 1$. Hence, requests for item $i$ from each group $g$ are generated according to a stationary process with rate $\lambda_i^g = \lambda^g b_i^g$.

Assume that the server is transmitting data according to some arbitrary broadcast schedule. Assume that each user from group $g$ has a local cache of size $K_g$ and the users of group $g$ are following an identical cache update strategy which determines the content of their caches. Let $A_i^g(n)$ be the total number of item $i$ requests generated by the users of group $g$ during slot $n$. Let $X_i^g(n)$ be the number of group $g$ user requests for item $i$ at the beginning of slot $n$. The item $i$ request backlog of group $g$ users evolves as follows:

$$X_i^g(n+1) = \begin{cases} 0 & if \;\; u_n = i \;\; or \;\; i \in C_g(n) \\ X_i^g(n) + A_i^g(n) & otherwise \end{cases}$$

where $C_g(n)$ is the set of $K_g$ items stored in the cache of group $g$ users during slot $n$.

Figure 3 shows the evolution of the expected item $i$ request backlog of group $g$ users, $\overline{X_i^g}(n)$, with caching (solid line) superimposed by the expected request backlog without caching (dotted line) for a given sequence of item $i$ broadcasts. The small rectangles above the horizontal time axis correspond to the slots at which item $i$ is broadcast. At the end of each item $i$ broadcast, all of the requests are granted. The slots at which item $i$ resides in the cache are represented by the small dashed rectangles below the time axis. All of the requests arriving during these slots are granted

immediately. The expected item $i$ request backlog of group $g$, $\overline{X_i^g}(s)$, starts increasing with rate $\lambda_i^g$ either from the end of the last item $i$ broadcast before $s$ or from the end of last item $i$ residence in the cache before $s$, whichever happened last. The aggregate expected delay of item $i$ requests with caching (without caching) is equal to the total area under the solid (dotted) curve. The area between solid and dotted curve represents the reduction in the aggregate expected delay of item $i$ requests due to caching.

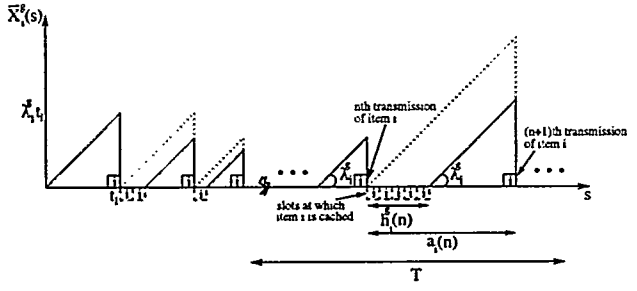


Figure 3: Expected item $i$ request backlog of group $g$ users with caching (solid line) and without caching (dotted line) as a function of time.

The first step for joint design of the server broadcast schedule and the user caching strategy is to lower-bound the average delay of item $i$ requests under a certain broadcast scheduling policy and under a certain cache management policy by the expression made up of average values of the component parameters. The minimum of the lowerbound over all possible values of these average parameters is a lowerbound of any broadcast scheduling and any cache management policy.

Consider an interval of duration $T$ as shown in figure 3. Let $N_i^T$ be the total number of transmissions of item $i$, $i = 1, ..., M$, during the interval $T$. Denote the inter-appearance gap between the $n$th transmission and $(n+1)$th transmission of item $i$ during the interval $T$ by $a_i(n)$. Assume that a user from group $g$ keeps item $i$ for $h_i^g(n)$ slots after the end of the $n$th transmission where $h_i^g(n) = 0, 1, ..., a_i(n)$ as shown in figure 3.

Consider the class of broadcast schedules where the limit of the average inter-appearance gap of each item $i$, $\overline{a_i}$, converges, which is defined as follows:

$$\overline{a_i} = \lim_{T\to\infty} \frac{1}{N_i^T} \sum_{n=1}^{N_i^T} a_i(n)$$

The fraction of time item $i$ is transmitted (average transmission frequency) in the schedule, $\overline{f_i}$, is equal to the inverse of its average inter-appearance gap and is given as follows:

$$\overline{f_i} = \frac{1}{\overline{a_i}} = \lim_{T\to\infty} \frac{N_i^T}{\sum_{n=1}^{N_i^T} a_i(n)} = \lim_{T\to\infty} \frac{N_i^T}{T}$$

Also consider the class of cache management policies where the limit of the average caching time of item $i$ by a user from group $g$, $\overline{h_i^g}$, exists, which is defined as follows:

$$\overline{h_i^g} = \lim_{T\to\infty} \frac{1}{N_i^T} \sum_{n=1}^{N_i^T} h_i^g(n)$$

The class of broadcast scheduling and cache management policies, for which the long-term average of the above parameters does not exist, is of little practical value and is of mathematical interest only.

The expected delay of item $i$ requests from group $g$ with caching between the $n$th and $(n+1)$th transmission of item $i$ is equal to the area of the solid line triangle between the $n$th and $(n+1)$th transmission of item $i$ shown in figure 3. Hence, the long-term average aggregate delay of item $i$ requests from group $g$, $D_i^g$, with caching is given by

$$D_i^g = \lim_{T\to\infty} \frac{1}{T} \sum_{n=1}^{N_i^T} \frac{\lambda_i^g}{2} (a_i(n) - h_i^g(n))^2$$

The following lemma gives the lower bound for $D_i^g$.

**Lemma 1** *The long-term average aggregate delay of item $i$ requests from group $g$, $D_i^g$, is lowerbounded by*

$$\frac{1}{2} \lambda_i^g \overline{f_i} \left(\overline{a_i} - \overline{h_i^g}\right)^2$$

**Proof:** Let $\overline{Y} = \frac{1}{N} \sum_{i=1}^{N} Y_i$.

$$\sum_{i=1}^{N} \left(Y_i - \overline{Y}\right)^2 \geq 0$$

$$\sum_{i=1}^{N} Y_i^2 - 2N\overline{Y}^2 + N\overline{Y}^2 \geq 0$$

Therefore,

$$\sum_{i=1}^{N} Y_i^2 \geq N\overline{Y}^2$$

If we apply it to our problem,

$$\lim_{T\to\infty} \frac{1}{T} \sum_{n=1}^{N_i^T} (a_i(n) - h_i^g(n))^2$$

$$\geq \lim_{T\to\infty} \frac{N_i^T}{T} \left\{ \frac{1}{N_i^T} \sum_{n=1}^{N_i^T} (a_i(n) - h_i^g(n)) \right\}^2$$

Since we assume the existence of the limits for the average parameters, the lemma is proved. ◇

The lowerbound of the long-term average aggregate delay of all the requests experienced by a user is given by

$$\frac{1}{2G} \sum_{g=1}^{G} \sum_{i=1}^{M} \lambda_i^g \overline{f_i} \left(\frac{1}{\overline{f_i}} - \overline{h_i^g}\right)^2$$

Consider the following optimization problem ($OP$).

$$OP: \quad D_{avg}^* = \min_{\overline{f_i}, \overline{h_i^g}, i=1,...,M, g=1,...,G} \frac{1}{2G} \sum_{g=1}^{G} \sum_{i=1}^{M} \lambda_i^g \overline{f_i} \left(\frac{1}{\overline{f_i}} - \overline{h_i^g}\right)^2 \quad (1)$$

$$\text{s.t.} \quad \sum_{i=1}^{M} \overline{f_i} \leq 1 \quad (2)$$

$$\sum_{i=1}^{M} \overline{f_i}\, \overline{h_i^g} \leq K_g, \quad g = 1, ..., G \quad (3)$$

The first constraint requires that the server cannot transmit more than one item in each slot and the second constraint says that we cannot store more items than the cache capacity of a user.

**Theorem 1** *The long-term average aggregate delay of all the requests experienced by a user under any broadcast scheduling and cache management policy is lowerbounded by the solution of the above optimization problem (OP).*

**Proof:** The minimum of the lowerbound over all possible values of these parameters is a lowerbound of any broadcast scheduling and any cache management policy. ◇

**Corollary 1** *For a given broadcast schedule with known average transmission frequencies $\overline{f_i}$'s, the long-term average aggregate delay of all the requests experienced by a user under any cache management strategy is lowerbounded by*

$$\frac{1}{2G}\sum_{g=1}^{G}\frac{(M-K_g)^2}{\sum_{i=1}^{M}\frac{\overline{f_i}}{\lambda_i^g}}.$$

**Proof:** When $\overline{f_i}$'s are known and we make full use of cache capacity, the optimization problem reduces to

$$\min_{\overline{h_i^g},i=1,...,M,g=1,...,G} \frac{1}{2G}\sum_{g=1}^{G}\sum_{i=1}^{M}\lambda_i^g\overline{f_i}\left(\frac{1}{\overline{f_i}}-\overline{h_i^g}\right)^2$$

$$\text{s.t.} \quad \sum_{i=1}^{M}\overline{f_i}\,\overline{h_i^g} = K_g, \quad g=1,...,G$$

By solving the optimization problem, the optimal average caching times are

$$\overline{h_i^g}^* = \frac{1}{\overline{f_i}} - \frac{1}{\lambda_i^g}\frac{(M-K_g)}{\sum_{i=1}^{M}\frac{\overline{f_i}}{\lambda_i^g}} \quad \text{for } i=1,...,M, g=1,...,G$$

and the optimal cost is

$$\frac{1}{2G}\sum_{g=1}^{G}\frac{(M-K_g)^2}{\sum_{i=1}^{M}\frac{\overline{f_i}}{\lambda_i^g}}.$$

◇

The bound derived in corollary 1 can be used as a benchmark to investigate of performance of a cache management policy. In the following, we will provide an approach to jointly schedule broadcast and do cache management using the parameters obtained from the above optimization problem (*OP*).

## 3 The Algorithm

One approach for joint broadcast scheduling and cache management is to construct a broadcast schedule according to the optimal transmission frequencies obtained from the optimization, $\overline{f_i}^*$, $i=1,...,M$, and to follow an optimal cache management policy [16] under the above-designed broadcast schedule.

Since the optimization problem (*OP*) is not convex, it is not guaranteed that the global optimal solution can be obtained. In section 4, numerical experiments are carried out to investigate the performance gain over separate design of broadcast schedule and cache management just by using the local optimal solutions obtained from the (*OP*).

### 3.1 Aggregate Access Statistics

Since usually there are more than one group with different user's access statistics, the scheduling algorithm transforms the set of user's access statistics into one aggregate access statistics which is used for scheduling as depicted in figure 2. There are a number of ways the scheduling policy can map the set of user's access statistics into one aggregate access statistics. One simple way is to take the average of all the access statistics and this is what most of the scheduling algorithms do. That is, the aggregate access probabilities are computed as follows:

$$b_i = \frac{1}{G}\sum_{g=1}^{G} b_i^g. \tag{4}$$

Joint broadcast scheduling and user's cache management suggests a novel approach for obtaining the aggregate user's access statistics. Under the joint approach, the optimal transmission frequencies $\overline{f_i}^*$'s are first obtained from the optimization and the aggregate user's access probabilities are computed from the $\overline{f_i}^*$'s as follows [5]:

$$b_i = \frac{\overline{f_i}^{*2}}{\sum_{j=1}^{M}\overline{f_j}^{*2}} \tag{5}$$

The results in section 4 show that we can achieve up to 40% improvement in mean response time by using joint approach over the use of the average of user's access statistics.

### 3.2 Broadcast Scheduling

#### 3.2.1 One-Level MAD

Among the algorithms proposed in literature to generate a broadcast schedule with transmission frequencies which are close to the given ones, the *MAD* [15] is an on-line low complexity algorithm which yields the mean response time of user's requests very close to the lower bound. Therefore, we consider the MAD for constructing broadcast schedules.

At each slot $n$, the MAD broadcast the item

$$u_n = \arg\max_{i=1,...,M} b_i\, w_i(n)^2$$

where $w_i(n)$ is the elapsed time until the beginning of slot $n$ since the last transmission of item $i$, as illustrated in figure 4. Ties can be broken arbitrarily.

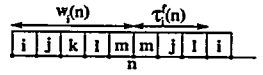


Figure 4: Illustration of $w_i(n)$ and $\tau_i^f(n)$

To generate broadcast schedules, the MAD algorithm requires user's access probabilities, $b_i$'s. The $b_i$'s can be obtained either from equation (5) or from equation (4) when there are more than one group with different access statistics.

Table 1 compares the given average transmission frequencies and those of the broadcast schedule generated by the MAD for 100 items. The MAD algorithm, indeed, constructs a broadcast schedule with average transmission frequencies which are very close to the required ones.

As the following example shows, it is also necessary to take the group identity of an item (which group of users is

Table 1: Given $\overline{f_i}$'s vs. $\overline{f_i}$'s produced by the MAD

| Item | Given $\overline{f_i}$ | MAD $\overline{f_i}$ | Item | Given $\overline{f_i}$ | MAD $\overline{f_i}$ |
|---|---|---|---|---|---|
| 1 | 0.001998 | 0.002040 | 51 | 0.009896 | 0.009754 |
| 2 | 0.015707 | 0.015543 | 52 | 0.009823 | 0.009754 |
| 3 | 0.018721 | 0.018655 | 53 | 0.009767 | 0.009754 |
| 4 | 0.007432 | 0.007510 | 54 | 0.009688 | 0.009754 |
| 5 | 0.011231 | 0.011283 | 55 | 0.009616 | 0.009754 |
| 6 | 0.013950 | 0.013832 | 56 | 0.009533 | 0.009736 |
| 7 | 0.015479 | 0.015324 | 57 | 0.009462 | 0.009736 |
| 8 | 0.011252 | 0.011283 | 58 | 0.009392 | 0.009736 |
| 9 | 0.008469 | 0.008424 | 59 | 0.009298 | 0.009461 |
| 10 | 0.005943 | 0.006008 | 60 | 0.009162 | 0.009201 |
| 11 | 0.016820 | 0.016731 | 61 | 0.008483 | 0.008424 |
| 12 | 0.013926 | 0.013832 | 62 | 0.009104 | 0.009200 |
| 13 | 0.012078 | 0.012023 | 63 | 0.009021 | 0.009091 |
| 14 | 0.011691 | 0.011547 | 64 | 0.008902 | 0.008924 |
| 15 | 0.013927 | 0.013832 | 65 | 0.008816 | 0.008828 |
| 16 | 0.012403 | 0.012352 | 66 | 0.008728 | 0.008724 |
| 17 | 0.013161 | 0.012977 | 67 | 0.008649 | 0.008633 |
| 18 | 0.014386 | 0.014333 | 68 | 0.008571 | 0.008536 |
| 19 | 0.014034 | 0.013832 | 69 | 0.008488 | 0.008424 |
| 20 | 0.013850 | 0.013832 | 70 | 0.008411 | 0.008424 |
| 21 | 0.013752 | 0.013832 | 71 | 0.008336 | 0.008290 |
| 22 | 0.013412 | 0.013261 | 72 | 0.008266 | 0.008186 |
| 23 | 0.013345 | 0.013261 | 73 | 0.008204 | 0.008136 |
| 24 | 0.013073 | 0.012977 | 74 | 0.008141 | 0.008062 |
| 25 | 0.012933 | 0.012976 | 75 | 0.008081 | 0.008011 |
| 26 | 0.012733 | 0.012685 | 76 | 0.008016 | 0.007860 |
| 27 | 0.012531 | 0.012352 | 77 | 0.007960 | 0.007850 |
| 28 | 0.012366 | 0.012352 | 78 | 0.007906 | 0.007850 |
| 29 | 0.012179 | 0.012023 | 79 | 0.007847 | 0.007708 |
| 30 | 0.012053 | 0.012023 | 80 | 0.007790 | 0.007614 |
| 31 | 0.011890 | 0.011785 | 81 | 0.007738 | 0.007613 |
| 32 | 0.011748 | 0.011548 | 82 | 0.007683 | 0.007511 |
| 33 | 0.011637 | 0.011547 | 83 | 0.007635 | 0.007511 |
| 34 | 0.011513 | 0.011296 | 84 | 0.007588 | 0.007511 |
| 35 | 0.011399 | 0.011284 | 85 | 0.007539 | 0.007510 |
| 36 | 0.011291 | 0.011283 | 86 | 0.007492 | 0.007510 |
| 37 | 0.011185 | 0.011283 | 87 | 0.007445 | 0.007510 |
| 38 | 0.011069 | 0.011134 | 88 | 0.007400 | 0.007510 |
| 39 | 0.010937 | 0.010931 | 89 | 0.007353 | 0.007510 |
| 40 | 0.010799 | 0.010749 | 90 | 0.007305 | 0.007510 |
| 41 | 0.010683 | 0.010625 | 91 | 0.007264 | 0.007510 |
| 42 | 0.010571 | 0.010507 | 92 | 0.007215 | 0.007470 |
| 43 | 0.010448 | 0.010341 | 93 | 0.007174 | 0.007469 |
| 44 | 0.010341 | 0.010185 | 94 | 0.007132 | 0.007469 |
| 45 | 0.010263 | 0.010185 | 95 | 0.007093 | 0.007469 |
| 46 | 0.010153 | 0.010008 | 96 | 0.007051 | 0.007469 |
| 47 | 0.010044 | 0.009755 | 97 | 0.007013 | 0.007469 |
| 48 | 0.009990 | 0.009755 | 98 | 0.006971 | 0.007469 |
| 49 | 0.010015 | 0.009755 | 99 | 0.006933 | 0.007469 |
| 50 | 0.009920 | 0.009755 | 100 | 0.006896 | 0.007469 |

interested in the item) into consideration when user's cache management is considered together with broadcast scheduling.

### 3.2.2 Two-Level MAD

Consider a case with 20 items and 10 groups. Each group has cache of size one and group $i$ is assumed to be interested only in two items, item $2i-1$ and $2i$ where $i = 1, ..., 10$ with equal probabilities. Since the average aggregate access probabilities are uniform, the optimal schedule is cyclic with each item broadcast once. Moreover, one-step look-ahead (OSLA) cache management policy is optimal as user's access probabilities of each group is uniform among the items of interest (Proof is similar to that of theorem 1 in [16]).

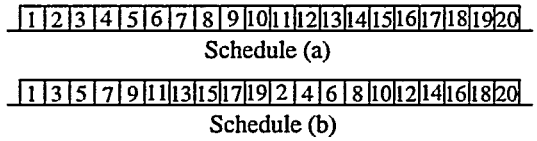


Figure 5: A pathological example to one-level scheduling

Consider two cyclic schedules as depicted in figure 5. Without caching, the mean response time experienced by each group will be the same under the two schedules. However, with the use of cache of size one and the optimal cache management policy, the mean response time experienced by each group under the schedule (a) and schedule (b) in figure 5 are 90.50 and 50.00 slots respectively. It shows that the items should be uniformly spaced according to group basis in addition to the uniform spacing among the items. Hence, group-level scheduling is also required to ensure the lower mean response time when caching is performed on user's side.

The MAD algorithm can be easily extended for group-level consideration. Let's call it "*two-level MAD*". The two-level MAD first selects the user group to which the item to broadcast belongs and then selects the item among those the selected group of users is interested in.

$$(1)\quad g_n = \arg\max_{g=1,...,G} \quad \sum_{i=1}^{M} b_i^g w_i(n)^2$$

$$(2)\quad u_n = \arg\max_{i=1,...,M} \quad b_i^{g_n} w_i(n)^2$$

Ties can be broken in an arbitrary manner.

It is obvious that the two-level MAD generates the "*good*" schedule for the pathological example to the one-level MAD. In section 4, we will compare the performance of the two-level MAD to that of the one-level MAD for a few examples.

### 3.3 Cache Management

Due to high implementation complexity, the optimal cache management provided in [16] is not practical. As the numerical examples in [16] showed that the low-complexity *One-Step Look-Ahead* (*OSLA*) strategy yields performance comparable to that of the optimal policy in many cases, we will apply the OSLA scheme for cache management in our numerical experiments in section 4. The OSLA policy computes the *reward of caching* for the items already stored in the cache and for the arriving item on the broadcast and discards the one with the smallest reward. The reward of caching item $i$ at slot $n$ is equal to the reduction in the expected delay of item $i$ requests due to the caching of item $i$ at slot $n$ and is given by

$$\lambda_i \tau_i^f(n) + \frac{\lambda_i}{2} \qquad (6)$$

where $\tau_i^f(n)$ is the number of slots from the beginning of slot $n$ until the beginning of the first slot after $n$ at which item $i$ is transmitted as illustrated in figure 4. In the following, we will give an informal proof for (6) (For details, interested readers are referred to [16]).

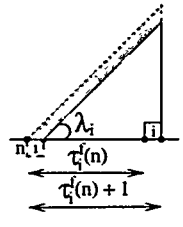


Figure 6: The reward of caching item $i$ at slot $n$

Assume that item $i$ is stored in the cache at slot $n$ and is discarded from the cache at the end of slot $n$ as shown in figure 6. Also assume that item $i$ is transmitted by the server at slot $n+\tau_i^f(n)$ the first time after $n$. The aggregate expected delay with caching (without caching) from slot $n$ to slot $n+\tau_i^f(n)+1$ is equal to the area of the solid (dotted) triangle. The shaded area between the dotted and solid

curve is equal to the reduction in the expected delay of item $i$ requests due to the caching of item $i$ during slot $n$. Therefore, if we subtract the area of the solid triangle from that of the dotted triangle, we obtain the reward of caching item $i$ at slot $n$ which turns out to be equal to (6).

## 4 Performance of the Joint Approach

In this section, the performance of our Joint Scheduling and Cache Management (JSCM) approach is compared to that of the Non-Joint Scheduling and Cache Management (Non-JSCM) scheme. The only difference between JSCM and Non-JSCM is user's access probabilities, $b_i$'s. In JSCM, $b_i$'s are computed from the average transmission frequencies, $\overline{f_i}$'s, by using equation (5), which are obtained from the optimization problem (OP). The optimization problem ($OP$) is solved by using the Feasible Sequential Quadratic Programming (FSQP) software [17]. In Non-JSCM, $b_i$'s are obtained by taking the average of the given access probabilities over different groups as in equation (4). The $b_i$'s are used as an input to the one-level and two-level MAD algorithms. Then, we employ the OSLA as our cache management strategy under the schedules generated by JSCM and Non-JSCM and compare the mean response time of two approaches.

Numerical experiments are performed for 100 items. Items are numbered from 1 to 100 and users from each group $g$ are assumed to be interested only in $M_g$ consecutive items starting from $Start\ Item_g$ up to $(Start\ Item_g + M_g - 1) \bmod (M + 1) + 1$. For these $M_g$ items, user's access probabilities are assumed to follow zipf distribution [12] where $\tilde{b}_j = \frac{j^\theta - (j-1)^\theta}{M^\theta}$ where $j = 1, ..., M_g$. As $\theta$ decreases, the access pattern becomes increasingly skewed. For $\theta = 1$, zipf distribution reduces to uniform distribution.

User's access probabilities of each group are either in *ascending* order or in *descending* order from item $Start\ Item_g$ to $(Start\ Item_g + M_g - 1) \bmod (M + 1) + 1$.
For the case with descending order,
if $(Start\ Item_g + M_g - 1) \leq M$, $b_i = \tilde{b}_{i - Start\ Item_g + 1}$.
Otherwise,

$b_i = \tilde{b}_{i - Start\ Item_g + 1}$
$i = Start\ Item_g, ..., M$
$b_i = \tilde{b}_{i + M - Start\ Item_g + 1}$
$i = 1, ..., (Start\ Item_g + M_g - 1) \bmod (M + 1) + 1$

For the case with ascending order,
if $(Start\ Item_g + M_g - 1) \leq M$, $b_i = \tilde{b}_{M_g - i + Start\ Item_g}$.
Otherwise,

$b_i = \tilde{b}_{M_g - i + Start\ Item_g}$
$i = Start\ Item_g, ..., M$
$b_i = \tilde{b}_{M_g - i - M + Start\ Item_g}$
$i = 1, ..., (Start\ Item_g + M_g - 1) \bmod (M + 1) + 1$

The examples are provided at the end of the paper. The numerical results in examples I to VIII compare the mean response time (MRT) of JSCM and Non-JSCM and the MRT of one-level and two-level MAD. The first and second big column shows the mean response time without user's caching and the mean response time reduced by caching with the OSLA strategy respectively. The reduction in the mean response time due to caching under any cache management strategy is equal to the aggregate sum of the reward of caching for the items stored in the cache at each slot over the one period of the broadcast schedule [16] and is given by

$$\sum_{n=1}^{T_s} \sum_{i \in C(n)} \lambda_i \left( \tau_i^f(n) + \frac{1}{2} \right)$$

where $T_s$ is the one period of the broadcast schedule and $C(n)$ is the set of items stored in the cache at slot $n$ as defined before.

The third column shows the actual mean response time experienced by a user with caching using the OSLA policy. This value is equal to the difference between MRT without caching (first column) and MRT reduced by caching (second column). The "*Optimized*" subcolumn gives the results under JSCM while the "*Average*" subcolumn shows the results for Non-JSCM. The last column shows the % improvement in the mean response time over Non-JSCM by the use of JSCM.

The number of groups considered in the experiment ranges from 1 to 5. Example I corresponds to the case with one group, examples II, III and IV to the case with two groups and examples V, VI, VII and VIII to the case with five groups. For multi-group cases, access patterns are considered for the following three cases: non-overlapping (different groups of users are interested in different sets of items with no common items of interest), partial overlapping (with some common items of interest to different groups) and complete overlapping (all the groups considered are interested in all the items with non-zero access probabilities). Example II and V are for complete overlapping case, III and VIII for non-overlapping case and IV, VI and VII for partial overlapping case. Different groups have access probabilities with different skewedness (different values of $\theta$).

With JSCM, MRT without caching is quite large but a large portion of MRT is later reduced by caching. Although MRT without caching for Non-JSCM is smaller than that for JSCM in all cases, caching reduces only a smaller portion of MRT for Non-JSCM than JSCM. Therefore, the improvement in the actual MRT experienced by a user with caching over Non-JSCM by JSCM ranges from 10% up to 40%. It shows that considerably better performance can be achieved if we design broadcast schedules which are more amenable to user's caching.

First let's look at the case with one-level MAD first. Even for the single-group case (example I) with cache size 10, 100 items and $\theta = \log(0.8)/\log(0.2)$, we can gain 18.10% improvement with JSCM. In the case with 5 groups (example VIII), the gain over Non-JSCM by JSCM is as high as 40.78%. One important observation is that we can reduce MRT more by employing JSCM when access probabilities of different groups are non-overlapping for both 2-group and 5-group cases (see examples III and VIII). For the case with 5 groups, the gain over Non-JSCM is smallest when all the groups are interested in all the items (complete overlapping). When we decrease the cache size (compare examples VI and VII), the improvement over Non-JSCM also decreases a little.

For JSCM, two-level MAD yields almost the same MRT as one-level MAD for all the cases. It shows that one-level scheduling performs well in most cases (except a few pathological cases). However, for Non-JSCM, the improvement in MRT over one-level by two-level MAD is larger. Therefore, one-level MAD is good enough to be a candidate for the low complexity implementation of JSCM.

## 5 Concluding Remarks

We consider the problem of joint broadcast scheduling and user's cache management such that the mean response time experienced by the users is minimized. Under our joint approach, broadcast schedules are designed such that the schedules are conducive to user's caching and the overall mean response time experienced by the users can be considerably reduced when caching is performed at user's side. We do gain improvement up to 40% over the traditional broadcast scheduling approach which does not take into account of user's caching. Although the computational complexity of solving the optimization problem is not trivial, our approach is amenable to implementation since the computation can be performed *off-line*. We also show an example that shows that it is also necessary to take the group identity of an item (which group of users is interested in the item) into consideration for scheduling when prefetching is done at user's side. We provide a two-level scheduling policy which is immune to pathological cases and performs slightly better than one-level scheduling in general. A valuable byproduct of our approach is a lowerbound for the performance of any cache management strategy when the server's broadcast schedule is known. It can be used as an benchmark to investigate the performance of a cache management policy.

## References


[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. "Broadcast Disks: Data Management for Asymmetric Communication Environments". Technical Report CS-94-43, Dept. of Comp. Science, Brown University, October 1994.

[2] S. Acharya, M. Franklin, and S. Zdonik. "Prefetching from a Broadcast Disk". In *Proc. 12th Int'l. Conf. Data Eng.*, New Orleans, LA, February 1996.

[3] AirMedia Inc. WWW. URL, http://www.airmedia.com/. 1997.

[4] M. H. Ammar. "Response Time in a Teletext System: an Individual User's Perspective". *IEEE Transaction on Communication*, COM-35(11):1159–1170, November 1987.

[5] M. H. Ammar and J. W. Wong. "The Design of Teletext Broadcast Cycles". *Performance Evaluation*, 5(4):235–242, December 1985.

[6] M. H. Ammar and J. W. Wong. "On the Optimality of Cyclic Transmission in Teletext Systems". *IEEE Transaction on Communication*, COM-35(1):68–73, January 1987.

[7] T. Bowen, G. Gopal, G. Herman, T. Hickey, K. Lee, W. Mansfield, J. Raitz, and A. Weinrib. "The Datacycle Architecture". *Communications of the ACM*, 35(12):71–81, December 1992.

[8] D. K. Gifford. "Polychannel Systems for Mass Digital Communication". *Communications of the ACM*, 33(2):141–151, February 1990.

[9] S. Hameed and N. H. Vaidya. Log-time Algorithms for Scheduling Single and Multiple Channel Data Broadcast. In *Proc. of MOBICOM'97*, Budapest, Hungary, September 1997.

[10] DirecPC. WWW Hughes Network Systems. URL, http://www.direcpc.com/. 1996.

[11] Intel Intercast Technology. WWW. URL, http://www.intercast.com/. 1997.

[12] Donald E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, Reading, Massachusetts, second edition, 1981.

[13] Inc. WWW Marimba. URL, http://www.marimba.com/. 1997.

[14] Pointcast Inc. WWW. URL, http://www.pointcast.com/. 1997.

[15] C. J. Su and L. Tassiulas. "Broadcast Scheduling for Information Distribution". In *Proc. IEEE INFOCOM'97*, volume I, pages 109–117, Kobe, Japan, 1997.

[16] L. Tassiulas and C. J. Su. "Optimal Memory Management Strategies for a Mobile User in a Broadcast Data Delivery System". *IEEE JSAC Special Issue on Networking and Performance Issues of Personal Mobile Communications*, 15(7):1226–1238, September 1997.

[17] Andri Tits. "FSQP". In *URL, http://www.isr.umd.edu/Labs/CACSE/FSQP/fsqp.html.*

[18] J. W. Wong. "Broadcast Delivery". *Proceedings of the IEEE*, 76(12):1566–1577, December 1988.

**Example I**
No. Of Groups = 1,
$K_1 = 10$, $M_1 = 100$, *Start Item*$_1 = 1$ and $\theta = \log(0.8)/\log(0.2)$ in *decreasing* order.

| | MRT Without Caching | | MRT Reduced by Caching | | MRT Actually Experienced | | % Improvement over the use of Average Parameters |
|---|---|---|---|---|---|---|---|
| | Optimized | Average | Optimized | Average | Optimized | Average | |
| Group 1 | 151.21 | 23.14 | 138.86 | 8.06 | 12.35 | 15.08 | 18.10 % |

**Example II**
No. Of Groups = 2,
$K_1 = 10$, $M_1 = 100$, *Start Item*$_1 = 1$ and $\theta = \log(0.8)/\log(0.2)$ in *decreasing* order.
$K_2 = 10$, $M_2 = 100$, *Start Item*$_2 = 1$ and $\theta = \log(0.8)/\log(0.2)$ in *increasing* order.

| | MRT Without Caching | | MRT Reduced by Caching | | MRT Actually Experienced | | % Improvement over the use of Average Parameters |
|---|---|---|---|---|---|---|---|
| | Optimized | Average | Optimized | Average | Optimized | Average | |
| Group 1 | 38.35 | 27.69 | 25.22 | 11.88 | 13.13 | 15.81 | 16.95 % |
| Group 2 | 126.93 | 27.69 | 114.13 | 11.88 | 12.80 | 15.81 | 19.04 % |
| Average | 82.64 | 27.69 | 69.68 | 11.88 | 12.96 | 15.81 | 18.03 % |

**Example III**
No. Of Groups = 2,
$K_1 = 10$, $M_1 = 50$, *Start Item*$_1 = 1$ and $\theta = \log(0.8)/\log(0.2)$ in *decreasing* order.
$K_2 = 10$, $M_2 = 50$, *Start Item*$_2 = 51$ and $\theta = \log(0.8)/\log(0.2)$ in *decreasing* order.

| | MRT Without Caching | | MRT Reduced by Caching | | MRT Actually Experienced | | % Improvement over the use of Average Parameters |
|---|---|---|---|---|---|---|---|
| | Optimized | Average | Optimized | Average | Optimized | Average | |
| Group 1 | 81.53 | 23.93 | 72.54 | 12.07 | 8.99 | 11.86 | 24.20 % |
| Group 2 | 80.14 | 23.93 | 71.22 | 12.07 | 8.92 | 11.86 | 24.79 % |
| Average | 80.84 | 23.93 | 71.88 | 12.07 | 8.96 | 11.86 | 24.45 % |

**Example IV**
No. Of Groups = 2,
$K_1 = 10$, $M_1 = 80$, *Start Item*$_1 = 1$ and $\theta = \log(0.8)/\log(0.2)$ in *increasing* order.
$K_2 = 10$, $M_2 = 80$, *Start Item*$_2 = 21$ and $\theta = \log(0.8)/\log(0.2)$ in *decreasing* order.

| | MRT Without Caching | | MRT Reduced by Caching | | MRT Actually Experienced | | % Improvement over the use of Average Parameters |
|---|---|---|---|---|---|---|---|
| | Optimized | Average | Optimized | Average | Optimized | Average | |
| Group 1 | 102.67 | 24.44 | 91.76 | 11.10 | 10.91 | 13.34 | 18.22 % |
| Group 2 | 30.93 | 24.44 | 19.75 | 11.10 | 11.18 | 13.34 | 16.19 % |
| Average | 66.80 | 24.44 | 55.76 | 11.10 | 11.05 | 13.34 | 17.17 % |

**Example V**
No. Of Groups = 5,
$K_1 = 5$, $M_1 = 100$, *Start Item*$_1 = 1$, $\theta = 1$.
$K_2 = 10$, $M_2 = 100$, *Start Item*$_2 = 1$, $\theta = 0.5$ and in *decreasing* order.
$K_3 = 15$, $M_2 = 100$, *Start Item*$_3 = 1$, $\theta = 0.1386$ and in *increasing* order.
$K_4 = 20$, $M_2 = 100$, *Start Item*$_4 = 1$, $\theta = 0.025$ and in *decreasing* order.
$K_5 = 25$, $M_2 = 100$, *Start Item*$_5 = 1$, $\theta = 0.005$ and in *increasing* order.

| | MRT Without Caching | | MRT Reduced by Caching | | MRT Actually Experienced | | % Improvement over the use of Average Parameters |
|---|---|---|---|---|---|---|---|
| | Optimized | Average | Optimized | Average | Optimized | Average | |
| Group 1 | 51.23 | 55.87 | 6.01 | 5.78 | 45.22 | 50.10 | 9.74% |
| Group 2 | 53.91 | 49.71 | 21.79 | 13.40 | 32.12 | 36.31 | 11.54% |
| Group 3 | 61.64 | 28.41 | 50.95 | 16.32 | 10.69 | 12.09 | 11.58% |
| Group 4 | 56.76 | 13.43 | 54.99 | 11.39 | 1.77 | 2.04 | 13.24% |
| Group 5 | 70.26 | 8.34 | 69.95 | 7.80 | 0.31 | 0.34 | 8.82% |
| Average | 58.76 | 31.15 | 40.74 | 10.94 | 18.02 | 20.18 | 10.70% |

**Example VI**
No. Of Groups = 5,
$K_1 = 5$, $M_1 = 40$, *Start Item*$_1 = 1$, $\theta = 0.025$ and in *decreasing* order.
$K_2 = 10$, $M_2 = 40$, *Start Item*$_2 = 21$, $\theta = 0.025$ and in *decreasing* order.
$K_3 = 15$, $M_2 = 40$, *Start Item*$_3 = 41$, $\theta = 0.025$ and in *decreasing* order.
$K_4 = 20$, $M_2 = 40$, *Start Item*$_4 = 61$, $\theta = 0.025$ and in *decreasing* order.
$K_5 = 25$, $M_2 = 40$, *Start Item*$_5 = 81$, $\theta = 0.025$ and in *decreasing* order.

**One-level MAD**

| | MRT Without Caching | | MRT Reduced by Caching | | MRT Actually Experienced | | % Improvement over the use of Average Parameters |
|---|---|---|---|---|---|---|---|
| | Optimized | Average | Optimized | Average | Optimized | Average | |
| Group 1 | 35.22 | 11.78 | 32.72 | 7.62 | 2.50 | 4.16 | 39.90% |
| Group 2 | 54.58 | 11.78 | 52.95 | 9.13 | 1.63 | 2.65 | 38.49% |
| Group 3 | 58.57 | 11.78 | 57.50 | 10.08 | 1.07 | 1.70 | 37.05% |
| Group 4 | 74.81 | 11.78 | 74.12 | 10.71 | 0.69 | 1.07 | 35.51% |
| Group 5 | 76.09 | 11.78 | 75.72 | 11.18 | 0.38 | 0.60 | 36.67% |
| Average | 59.85 | 11.78 | 58.60 | 9.74 | 1.25 | 2.04 | 38.73% |

**Two-level MAD**

| | MRT Without Caching | | MRT Reduced by Caching | | MRT Actually Experienced | | % Improvement over the use of Average Parameters |
|---|---|---|---|---|---|---|---|
| | Optimized | Average | Optimized | Average | Optimized | Average | |
| Group 1 | 35.47 | 11.89 | 32.99 | 7.88 | 2.49 | 4.02 | 38.06% |
| Group 2 | 54.84 | 11.89 | 53.23 | 9.35 | 1.61 | 2.54 | 36.61% |
| Group 3 | 57.91 | 11.89 | 56.84 | 10.26 | 1.07 | 1.63 | 34.36% |
| Group 4 | 74.14 | 11.89 | 73.45 | 10.87 | 0.69 | 1.02 | 32.35% |
| Group 5 | 75.81 | 11.89 | 75.44 | 11.33 | 0.37 | 0.56 | 33.93% |
| Average | 59.63 | 11.89 | 58.39 | 9.94 | 1.25 | 1.96 | 36.22% |

**Example VII**
No. Of Groups = 5,
$K_1 = 2$, $M_1 = 40$, *Start Item*$_1 = 1$, $\theta = 0.025$ and in *decreasing* order.
$K_2 = 4$, $M_2 = 40$, *Start Item*$_2 = 21$, $\theta = 0.025$ and in *decreasing* order.
$K_3 = 6$, $M_2 = 40$, *Start Item*$_3 = 41$, $\theta = 0.025$ and in *decreasing* order.
$K_4 = 8$, $M_2 = 40$, *Start Item*$_4 = 61$, $\theta = 0.025$ and in *decreasing* order.
$K_5 = 10$, $M_2 = 40$, *Start Item*$_5 = 81$, $\theta = 0.025$ and in *decreasing* order.

**One-level MAD**

| | MRT Without Caching | | MRT Reduced by Caching | | MRT Actually Experienced | | % Improvement over the use of Average Parameters |
|---|---|---|---|---|---|---|---|
| | Optimized | Average | Optimized | Average | Optimized | Average | |
| Group 1 | 30.73 | 11.78 | 27.17 | 6.14 | 3.57 | 5.64 | 36.70% |
| Group 2 | 31.65 | 11.78 | 28.82 | 7.21 | 2.83 | 4.57 | 38.07% |
| Group 3 | 33.23 | 11.78 | 30.90 | 7.98 | 2.33 | 3.80 | 38.68% |
| Group 4 | 32.56 | 11.78 | 30.60 | 8.61 | 1.96 | 3.17 | 38.17% |
| Group 5 | 37.31 | 11.78 | 35.66 | 9.13 | 1.65 | 2.65 | 37.74% |
| Average | 33.10 | 11.78 | 30.63 | 7.81 | 2.47 | 3.97 | 37.78% |

**Two-level MAD**

| | MRT Without Caching | | MRT Reduced by Caching | | MRT Actually Experienced | | % Improvement over the use of Average Parameters |
|---|---|---|---|---|---|---|---|
| | Optimized | Average | Optimized | Average | Optimized | Average | |
| Group 1 | 31.05 | 11.89 | 27.50 | 6.42 | 3.56 | 5.47 | 34.92% |
| Group 2 | 31.90 | 11.89 | 29.07 | 7.47 | 2.83 | 4.42 | 35.97% |
| Group 3 | 33.56 | 11.89 | 31.24 | 8.23 | 2.32 | 3.66 | 36.61% |
| Group 4 | 32.84 | 11.89 | 30.90 | 8.84 | 1.95 | 3.05 | 36.07% |
| Group 5 | 37.48 | 11.89 | 35.85 | 9.35 | 1.63 | 2.54 | 35.83% |
| Average | 33.37 | 11.89 | 30.91 | 8.06 | 2.46 | 3.83 | 35.77% |

**Example VIII**
No. Of Groups = 5,
$K_1 = 2$, $M_1 = 20$, *Start Item*$_1 = 1$, $\theta = 0.025$ and in *decreasing* order.
$K_2 = 4$, $M_2 = 20$, *Start Item*$_2 = 21$, $\theta = 0.025$ and in *decreasing* order.
$K_3 = 6$, $M_2 = 20$, *Start Item*$_3 = 41$, $\theta = 0.025$ and in *decreasing* order.
$K_4 = 8$, $M_2 = 20$, *Start Item*$_4 = 61$, $\theta = 0.025$ and in *decreasing* order.
$K_5 = 10$, $M_2 = 20$, *Start Item*$_5 = 81$, $\theta = 0.025$ and in *decreasing* order.

**One-level MAD**

| | MRT Without Caching | | MRT Reduced by Caching | | MRT Actually Experienced | | % Improvement over the use of Average Parameters |
|---|---|---|---|---|---|---|---|
| | Optimized | Average | Optimized | Average | Optimized | Average | |
| Group 1 | 25.21 | 10.49 | 22.47 | 5.90 | 2.74 | 4.59 | 40.31% |
| Group 2 | 28.49 | 10.49 | 26.40 | 6.70 | 2.08 | 3.49 | 40.40% |
| Group 3 | 31.47 | 10.49 | 29.91 | 7.83 | 1.56 | 2.66 | 41.35% |
| Group 4 | 29.76 | 10.49 | 28.61 | 8.51 | 1.16 | 1.97 | 41.12% |
| Group 5 | 28.46 | 10.49 | 27.64 | 9.09 | 0.82 | 1.40 | 41.43% |
| Average | 28.68 | 10.49 | 27.01 | 7.61 | 1.67 | 2.82 | 40.78% |

**Two-level MAD**

| | MRT Without Caching | | MRT Reduced by Caching | | MRT Actually Experienced | | % Improvement over the use of Average Parameters |
|---|---|---|---|---|---|---|---|
| | Optimized | Average | Optimized | Average | Optimized | Average | |
| Group 1 | 25.69 | 10.73 | 22.95 | 6.38 | 2.73 | 4.35 | 37.24% |
| Group 2 | 29.13 | 10.73 | 27.07 | 7.47 | 2.07 | 3.27 | 36.70% |
| Group 3 | 31.59 | 10.73 | 30.06 | 8.27 | 1.53 | 2.46 | 37.80% |
| Group 4 | 30.66 | 10.73 | 29.54 | 8.92 | 1.12 | 1.81 | 38.12% |
| Group 5 | 28.71 | 10.73 | 27.93 | 9.47 | 0.79 | 1.26 | 37.30% |
| Average | 29.16 | 10.73 | 27.51 | 8.10 | 1.65 | 2.63 | 37.26% |