An Efficient Scheduling Method for Query-Set-based Broadcasting in Mobile Environments*

Ye-In Chang and Wu-Han Hsieh
Dept. of Computer Science and Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan, R.O.C
E-mail: changyi@cse.nsysu.edu.tw

Abstract

Most of the previous researches assume that each mobile client needs only one data item. However, in many situations, a mobile client might need more than one data item. In this paper, we propose an efficient scheduling method for query-set-based broadcasting, which integrates with Query Expansion Method (QEM) and the mining association rules technique. The mining association rules can globally find the data item sets (large itemsets) which are requested by clients frequently. From our simulation results, we show that, as compared to the local optimal approach in the previous methods, our Improved-QEM can construct the schedule with the smaller TQD than QEM and Modified-QEM, where TQD denotes Total Query Distance and is proportional to the average access time.

1 Introduction

In the evolving field of mobile computing, there is a growing concern to provide mobile users with timely access to large amounts of information [3]. The communication asymmetric which means that the bandwidth in the downstream direction (servers-to-clients) is much greater than that in the upstream direction, along with the restriction in power that the mobile units have, make the model of broadcasting data to the clients, an attractive proposition. In wireless computing environments, to use the broadcasting mode not only can save the bandwidth of the uplink channel and but also can be scale to any number of mobile clients who listen to the publishing report.

Under the broadcasting mode, the server must construct a broadcast schedule to meet the needs of the client population [1]. There have been many strategies proposed for efficient broadcast delivery [4, 6, 7, 9, 10]. Most of the previous approaches assume that each mobile client needs only one data item. However, in many situations, a mobile client might need data of more than one item. For example, a mobile client wants to know the stock prices of Cisco, Microsoft, and IBM at the same time. In the previous research, this mobile client needs to issue three queries to acquire these three stock prices individually. Besides, the server schedules data items without considering the relationship between them, which extends the *access time* to process the client's requests, where the *access time* means the amount of time which a client has to wait for the requested data items.

For example, given a broadcast schedule $<d_1, d_2, d_3, d_4, d_5>$ and a query q_1 retrieving data items d_1 and d_4, the client has to wait for three more time slots to access data item d_4 after reading data item d_1. However, if the broadcast schedule is formed as $< d_2, d_3, d_1, d_4, d_5 >$, then the client can retrieve data items d_1, d_4 in successive time slots. Basically, the closer the data items in the same query q_i are scheduled, the smaller the $QD(q_i)$ is, where $QD(q_i)$ denotes the *Query Distance* of query q_i which means the longest distance between any two data items in the same query q_i and is proportional to the average access time [5]. (Note that we have $QD(q_1) = 2$ in the second schedule now.) Therefore, the issue of scheduling the broadcast data for the situation that each client may access multiple data items can not be simply considered as multiple subissues, each scheduling the broadcast data for the situation that each client only retrieves one data item. If the size of a query and the number of queries grow huge, the broadcast scheduling will become a critical issue [5]. In this paper, we use

*This research was supported in part by the National Science Council of Republic of China under Grant No. NSC-89-2218-E-110-004.

the *Total Query Distance* (TQD) [5] which equals to $\sum_{q_i \in Q} QD(q_i) * freq(q_i)$ to evaluate the performance of the schedule, where Q is the set of queries requested by clients and $freq(q_i)$ is the relative frequency of query q_i.

In recent years, there have been some methods proposed to support query-set-based data broadcast scheduling, for example, *Query Expansion Method* (QEM) [5]. QEM constructs a broadcast schedule by expanding *Query Data Set* (QDS) of each query in a greedy manner after sorting the queries based on their frequencies. Greedy methods sometimes provide local optimal solutions, but not global solutions. Another example is the *Modified Query Expansion Method* (*Modified-QEM*) [8] which has released some restrictions in QEM. However, in some cases, the moving operation of *Modified-QEM* will not benefit the TQD. Therefore, in this paper, to provide an efficient scheduling for query-set-based broadcasting, we propose the *Improved QEM method* (*Improved-QEM*) which applies one of the data mining techniques, mining association rules, to provide a much better global view of those queries than the previous methods, resulting in a better performance. From our simulation results, we show that, as compared to the local optimal approach in the previous methods, our Improved-QEM can construct the schedule with the smaller TQD than QEM and Modified-QEM.

The rest of the paper is organized as follows. Section 2 gives a survey of QEM. Section 3 presents the basic idea. Section 4 presents the proposed method. In Section 5, we study the performance. Finally, Section 6 gives the conclusions.

2 A Survey of QEM

The policies of QEM [5] are as the follows. First, the higher-frequency query takes precedence over the lower-one when expanding the schedule. Second, when expanding a query, $QD(q_i)$ of query q_i which had been previously expanded remain unchanged. Finally, when expanding the QDS (Query Data Set) of query q_i into currently constructed schedule, QEM always minimizes $QD(q_i)$ as many as possible.

For example, we have $q_1 = \{d_1, d_2, d_4, d_5\}$, $q_2 = \{d_4, d_5, d_6, d_7, d_8\}$, $q_3 = \{d_2, d_3, d_5, d_6\}$, $freq(q_1) = 3$, $freq(q_2) = 2$ and $freq(q_3) = 1$. Initially, the schedule σ^{step0} is empty. According to *Policy* 1, the method finds the highest frequency query, q_1, and expands its QDS. Then, schedule σ^{step1} is formed as follows: $\sigma^{step1} = [d_1, d_2, d_4, d_5]$, where the data item s between the symbols '[' and ']' can be interchangeable.

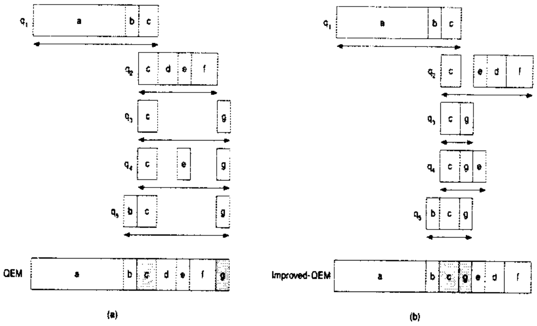


Figure 1: Motivation: (a) a schedule constructed by QEM; (b) a schedule constructed by *Improved-QEM*.

Second, the method expands query q_2. Since the current schedule σ^{step1} contains the data items d_4 and d_5 that are also in $QDS(q_2)$, the schedule is expanded into one of these forms:

$$\begin{aligned}\sigma^{step2}_{RightAppend} &= [d_1, d_2][d_4, d_5][d_6, d_7, d_8]\\ \sigma^{step2}_{LeftAppend} &= [d_6, d_7, d_8][d_4, d_5][d_1, d_2]\end{aligned}$$

Both of them give the same TQD value. In this example, we choose the former one for further expanding process. The schedule σ^{step2} minimizes $QD(q_2)$ (*Policy* 3), while preserving $QD(q_1)$ unchanged.

Finally, the $QDS(q_3)$ is expended. Among the data items in $QDS(q_3)$, only data item d_3 is not included in the current schedule σ^{step2}. To insert it into the schedule increases $QD(q_1)$ and (or) $QD(q_2)$, which violates *Policy* 2 of this method. Hence data item d_3 has to be appended to the left or right end of σ^{step2}.

When appending data item d_3, the data items d_2, d_5 and d_6 have to be moved (if allowed) for minimize $QD(q_3)$ like follows.

$$\begin{aligned}\sigma^{step3}_{LeftAppend} &= [d_3][d_1, d_2][d_4, d_5][d_6][d_7, d_8]\\ \sigma^{step3}_{RightAppend} &= [d_1][d_2][d_4, d_5][d_6, d_7, d_8][d_3]\end{aligned}$$

In the above two schedules, $\sigma^{step3}_{LeftAppend}$ results smaller TQD (= 4*3 + 5*2 + 7*1 = 29) than $\sigma^{step3}_{RightAppend}$.

3 The Basic Idea

Now, we use an example with related data as shown in Table 1 to show our basic idea, where itemset $a = \{1, 2, 3, 4, 5, 6, 7\}$, $b = \{8\}$, $c = \{9, 10\}$ and so on. (That is, the data set of query q_1

Table 1: Queries and the related data sets and frequencies

query	data set	frequency
q_1	{a,b,c}	5
q_2	{c,d,e,f}	4
q_3	{c,g}	3
q_4	{c,e,g}	2
q_5	{b,c,g}	1

Table 2: Large ItemSets

large itemsets	support (%)
{b,c}	40.0
{c,e}	40.0
{c,g}	40.0
{a,b,c}	33.3
{c,d,e,f}	26.7
{c,e,g}	13.3
{b,c,g}	6.7

contains {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}.) Figure 1-(a) shows the the schedule constructed by QEM. We observe that, if we can schedule the data items which are usually requested together at the same time as close as possible, the TQD of the schedule may be decreased. In Figure 1-(a), itemsets c and g are contained in three queries q_3, q_4 and q_5. If itemsets c and g can be scheduled closely, it may make many queries with small QD's. Based on this observation, we are motivated to apply the *mining association rules* technique to help us to find the item sets that clients request at the same time frequently. In some other view, we try to provide a global optimal approach to the query-set-based broadcast scheduling, as compared to the local optimal approach in the previous methods [5, 8].

In *mining association rules*, let I be a set of m distinct items [2]. A set of items is called an *itemset*. Itemsets of some length k are referred to as k-itemsets. A transaction T is said to *support* an itemset $X \subseteq I$ if it contains all items of X, i.e., $X \subseteq T$. The fraction of the transactions in the database that support X is called the *support* of X, denoted as *support*(X). An itemset is *large* if its support is above some user-defined minimum support threshold. Our basic idea is to find some useful large itemsets and replace the queries by these large itemsets. For the example shown in Figure 1, we find some useful large itemsets with related supports in Table 2. The large itemset $\{c, g\}$ is with a high support and will be scheduled first. We expand the schedule by these large itemsets and the result is shown in Figure 1-(b).

4 Improved-QEM

4.1 Assumptions

To simplify our algorithm, we made the following assumptions: (1) Clients' access patterns do not change. (2) Data is read-only. (3) Clients retrieve data items from the broadcast on demand. (4) Clients are simple and without a great amount of memory. (5) Clients make no use of their upstream communications capability. (6) When a client switches to the public channel, the client can retrieve data items immediately. (7) The server broadcasts data items over a single channel. (8) The broadcast infrastructure is reliable. (9) The length of each data item can be different. (10) A query can contain more than one data item. There is no precedence relationship in accessing data items. (11) Each query can be with the different frequency. (12) Each data item is broadcasted once within each broadcast cycle.

4.2 The Method

Basically, the steps of *Improved-QEM* are as follows:

Step 1: Apply the *mining association rules* technique to obtain the large itemsets with *supports* > 0 of queries and insert these large itemsets to *LargeItemset_list*. The large itemsets must be sorted by the number of data items in an increasing order.

Step 2: Prune the large itemsets with a single item from *LargeItemset_list* to *SingleItem_list*.

Step 3: Call procedure *Prune_Large_Itemset* as shown in Figure 2 to prune some large itemsets that are not necessary to be scheduled. Basically, if *LargeItemset*$(i) \subset$ *LargeItemset*(j) and *support*(i) = *support*(j), *LargeItemset*(i) is pruned.

Step 4: Sort the large itemsets in *LargeItemset_list* by the relative *support* in a decreasing order.

Step 5: Apply QEM to expand the schedule by the large itemsets in *LaregeItemset_list*, sequentially.

Step 6: Apply QEM to expand the schedule by the large itemsets in *SingleItem_list*.

```
prune_large_itemset()
 { for(int i=0; i < length of $LargeItemset\_list$; i++)
   {for(int j=1; i < i; j++)
    {if(($Set(j) \subset Set(i)$) and ($Support(j) = Support(i)$))
    {prune the large itemset $LargeItemset(j)$;
   j--;}}}}
```

Figure 2: Procedure *Prune_Large_Itemset*

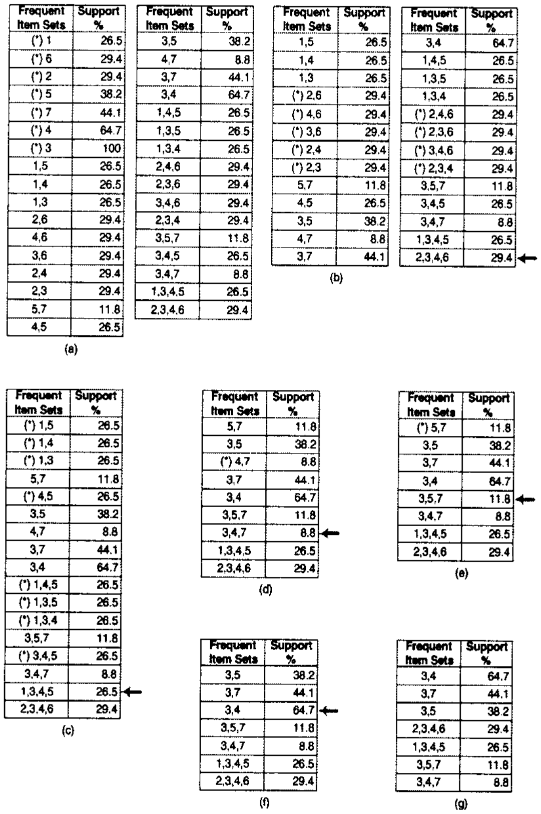
Table 3: Queries and the relative occurrence frequencies

query	query set	frequency
q_1	d_2, d_3, d_4, d_6	10
q_2	d_1, d_3, d_4, d_5	9
q_3	d_3, d_7	8
q_4	d_3, d_5, d_7	4
q_5	d_3, d_4, d_7	3

For the data as shown in Table 3, Figure 3 shows the related steps, where the rows marked with (*) are the large itemsets that will be pruned. In *Step* 1, since we want to find the data items that are requested simultaneously and frequently, we apply the ***mining association rules*** technique to carry out this work. We get the result of *LargeItemset_list* as shown in Figure 3-(a).

In *Step* 2, we do not consider the large itemsets with a single item and we prune them from *LargeItemset_list* to *SingleItem_list*. In Figure 3-(a), the marked rows will be pruned. The result of *LargeItemset_list* is shown in Figure 3-(b).

In *Step* 3, we continue to prune some unnecessary large itemsets, since there may exist some large itemsets X that are subsets of some other large itemsets Y, i.e., $X \subset Y$. For example, in Figure 3-(b), we obtain a large itemset $\{d_2, d_3, d_4, d_6\}$ (with support = 29.4%). The subset of it, for example, $\{d_2, d_6\}$ (with support = 29.4%) will also be obtained and could be pruned under the condition as stated in procedure *Prune_Large_Itemset*. Since under this condition, the positions in the schedule of data items d_2, and d_6 can be interchangeable, which will not change any $QD(q_i)$. In other words, if the data itemset $\{d_2, d_3, d_4, d_6\}$ is scheduled, we do not have to care about the positions in the schedule of data itemset $\{d_2, d_6\}$. However, the large itemset $\{d_3, d_4\}$ (with support = 64.7%) can not be pruned, since we observe that data items d_3 and d_4 must be scheduled as close as possible, such that it may make $QD(q_2)$ and $QD(q_5)$ as small as possible. Following this policy, Figure 3-(b) shows the result after considering the subsets of large itemset $\{d_2, d_3, d_4, d_6\}$. Next, in Figure 3-(c), we consider the subsets of the large itemset $\{d_1, d_3, d_4, d_5\}$. Similar to the previous step, we prune the marked large itemsets from *LargeItemset_list*. The processes in Figures 3-(d), 3-(e), and 3-(f) are the same as the previous two steps.



Frequent Item Sets	Support %
(*) 1	26.5
(*) 6	29.4
(*) 2	29.4
(*) 5	38.2
(*) 7	44.1
(*) 4	64.7
(*) 3	100
1,5	26.5
1,4	26.5
1,3	26.5
2,6	29.4
4,6	29.4
3,6	29.4
2,4	29.4
2,3	29.4
5,7	11.8
4,5	26.5

Frequent Item Sets	Support %
3,5	38.2
4,7	8.8
3,7	44.1
3,4	64.7
1,4,5	26.5
1,3,5	26.5
1,3,4	26.5
2,4,6	29.4
2,3,6	29.4
3,4,6	29.4
2,3,4	29.4
3,5,7	11.8
3,4,5	26.5
3,4,7	8.8
1,3,4,5	26.5
2,3,4,6	29.4

(a)

Frequent Item Sets	Support %
1,5	26.5
1,4	26.5
1,3	26.5
(*) 2,6	29.4
(*) 4,6	29.4
(*) 3,6	29.4
(*) 2,4	29.4
(*) 2,3	29.4
5,7	11.8
4,5	26.5
3,5	38.2
4,7	8.8
3,7	44.1

Frequent Item Sets	Support %
3,4	64.7
1,4,5	26.5
1,3,5	26.5
1,3,4	26.5
(*) 2,4,6	29.4
(*) 2,3,6	29.4
(*) 3,4,6	29.4
(*) 2,3,4	29.4
3,5,7	11.8
3,4,5	26.5
3,4,7	8.8
1,3,4,5	26.5
2,3,4,6	29.4 ←

(b)

Frequent Item Sets	Support %
(*) 1,5	26.5
(*) 1,4	26.5
(*) 1,3	26.5
5,7	11.8
(*) 4,5	26.5
3,5	38.2
4,7	8.8
3,7	44.1
3,4	64.7
(*) 1,4,5	26.5
(*) 1,3,5	26.5
(*) 1,3,4	26.5
3,5,7	11.8
(*) 3,4,5	26.5
3,4,7	8.8
1,3,4,5	26.5 ←
2,3,4,6	29.4

(c)

Frequent Item Sets	Support %
5,7	11.8
3,5	38.2
(*) 4,7	8.8
3,7	44.1
3,4	64.7
3,5,7	11.8
3,4,7	8.8 ←
1,3,4,5	26.5
2,3,4,6	29.4

(d)

Frequent Item Sets	Support %
(*) 5,7	11.8
3,5	38.2
3,7	44.1
3,4	64.7
3,5,7	11.8 ←
3,4,7	8.8
1,3,4,5	26.5
2,3,4,6	29.4

(e)

Frequent Item Sets	Support %
3,5	38.2
3,7	44.1
3,4	64.7 ←
3,5,7	11.8
3,4,7	8.8
1,3,4,5	26.5
2,3,4,6	29.4

(f)

Frequent Item Sets	Support %
3,4	64.7
3,7	44.1
3,5	38.2
2,3,4,6	29.4
1,3,4,5	26.5
3,5,7	11.8
3,4,7	8.8

(g)

Figure 3: The processing of *Improved-QEM*

In *Step* 4 and *Step* 5, we map the large itemsets and the relative supports to queries and the relative frequencies, respectively. Then, we use *QEM* to expand the schedule by the large itemsets in the order of the relative supports. After sorting the large itemsets by the relative *supports* in *LargeItemset_list*, Figure 3-(g) shows the result of the *LargeItemset_list*. We expand the schedule by these large itemsets in the order of the relative *supports* as follows.

(1) By $\{d_3, d_4\}$: $\sigma_1 = [d_3, d_4]$.

(2) By $\{d_3, d_7\}$: $\sigma_2 = [d_4][d_3][d_7]$.

(3) By $\{d_3, d_5\}$: $\sigma_3 = [d_4][d_3][d_7][d_5]$.

(4) By $\{d_2, d_3, d_4, d_6\}$: $\sigma_4 = [d_2, d_6][d_4][d_3][d_7][d_5]$.

(5) By $\{d_1, d_3, d_4, d_5\}$: $\sigma_5 = [d_2, d_6][d_4][d_3][d_7][d_5][d_1]$.

(6) By $\{d_3, d_5, d_7\}$: $\sigma_6 = [d_2, d_6][d_4][d_3][d_7][d_5][d_1]$.

(7) By $\{d_3, d_4, d_7\}$: $\sigma_7 = [d_2, d_6][d_4][d_3][d_7][d_5][d_1]$.

In *Step* 6, since we prune the large items with a single data item in *Step* 2, we must expand the schedule by considering any missing data item. In this example, there is no missing data item in σ_7, which is also the final result $\sigma_{Improved-QEM}$.

For the same example shown in Table 3, the schedules constructed by *QEM* and *Modified-QEM* are as follows:

$\sigma_{QEM} = [d_2, d_6][d_4][d_3][d_1, d_5][d_7]$.

$\sigma_{Modified-QEM} = [d_2, d_6][d_4][d_1][d_5][d_3][d_7]$.

In this example, the TQD's of *QEM*, *Modified-QEM*, and *Improved-QEM* are 139, 129 and 122, respectively. Therefore, *Improved-QEM* can construct a schedule that provides the smallest *TQD* among these methods.

5 Performance Study

5.1 The Performance Model

We generate synthetic query sets to evaluate the performance of *QEM*, *M-QEM* (i.e., *Modified-QEM*), and *I-QEM* (i.e., *Improved-QEM*) over a range of data characteristics. The parameters used in the generation of synthetic query sets contain D, P, $MinQ$, $MaxQ$, MRS, SRS, HF, NQ, and R. Parameter D is defined as the number of data items in the database. The database can be divided into P parts. Since we want to probe the effect of the relation of data itemsets that are requested at the same time, we define two special sets that are requested frequently, MRS and SRS. MRS is similar to the traffic information of the middle part of a super highway (which is requested most frequently), and SRS is similar to the traffic information of the substituted roads (which is requested a little less frequently than MRS). The value of $MinQ$ is defined as the sum of the length of MRS and SRS, while the value of $MaxQ$ is user defined. The length of a query is uniformly distributed from $MinQ$ to $MaxQ$. Parameter NQ is defined as the total number of queries to be scheduled. The highest frequency f_1 of the queries is determined by HF. Parameter R (the ratio of the frequencies of the two successive queries) is used to generate the frequency of the next query. If the frequency of the query q_i is f_i, we let $f_i = \lceil R * f_{i-1} \rceil$, where $2 \leq i \leq NQ$. Queries generated will satisfy the following conditions: (1) all queries are distinct. (2) the two highest frequencies of queries have to contain MRS, but not contain SRS; (3) other queries contain both MRS and SRS; (4) all other data items in a query are uniformly distributed in all partitions; (6) all data items in a query are distinct.

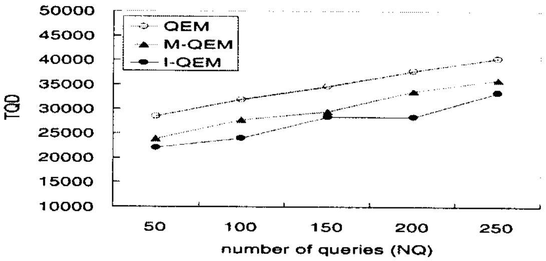


Figure 4: A comparison of the TQD (under the change of NQ)

For example, suppose $D = 20$, $MaxQ = 4$, $MinQ = 2$, $NQ = 5$, HF $(f_1) = 5$, $R = 0.5$. The length of each query is 4, 4, 2, 3, and 3, respectively. The two special data itemsets are $MRS = \{3\}$ and $SRS = \{7\}$. The frequencies of the queries q_2, q_3, q_4, and q_5 are 3 $(= \lceil 5 * 0.5 \rceil)$, 2 $(= \lceil 3 * 0.5 \rceil)$, 1 $(= \lceil 2 * 0.5 \rceil)$, and 1 $(= \lceil 1 * 0.5 \rceil)$, respectively. Queries q_1 and q_2 contain data item d_3, but do not contain data item d_7. All other queries (q_3, q_4, and q_5) contain both the data items d_3 and d_7. One of the possible cases of the generated queries contains $q_1 = \{d_2, d_3, d_4, d_5\}$, $q_2 = \{d_1, d_3, d_4, d_5\}$, $q_3 = \{d_3, d_7\}$, $q_4 = \{d_3, d_5, d_7\}$, and $q_5 = \{d_3, d_4, d_7\}$.

5.2 Simulation Results

For the simulation results presented here, the size of database (D) is 500. Similar results can be obtained for larger values as well. To simplify our concerns, we assign the values of other parameters $MaxQ$, $MinQ$, $|MRS|$, $|SRS|$, HF, and P to 5, 2, 1, 1, 500, and 2, respectively. In Figure 4, we show that our *I-QEM* provides better performance of the *TQD* than the other two methods, where $R = 0.75$. This is because our *I-QEM* can extract the data itemsets which are contained in most queries and schedule them as closer as possible in the broadcast schedule. As the number of the queries is increased, the *TQD*'s of the three methods are increased. In Figure 5,

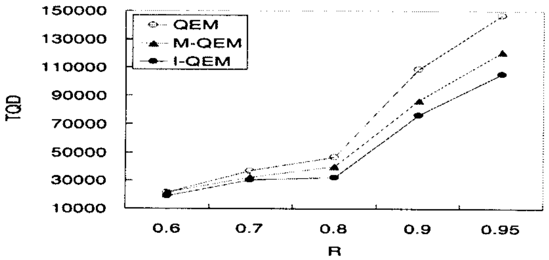


Figure 5: A comparison of the TQD (under the change of parameter R)

we also show that our *I-QEM* provides better performance of the TQD than the other two methods, where $NQ = 125$. This is because when the the value of the parameter R is large, it means that, the frequencies of the queries are decreasing very steady. Based on the same reason explained before, as the value of the parameter R is increased, the TQD's of these three methods are increased. Consequently, if most of the queries contain some common data items, it is better to schedule the data items that are contained in more queries before the data items that are contained in fewer queries.

6 Conclusion

In this paper, we have presented an improved version of QEM to improve the performance of QEM and *Modified-QEM*. Our basic idea is to find large itemsets. If we expand the schedule by QEM with large itemsets and expand the schedule in the order from a large itemset with a high support to a large itemset with a low support, we can construct a better schedule with the lower TQD than the previous methods. From our simulation results, we have shown that, our *Improved-QEM* can construct the schedule with the smaller TQD than QEM and *Modified-QEM*. How to schedule the query-set-based requests in the environment with multiple broadcast channels is the possible future work.

References

[1] S. Acharya, M. Franklin, S. Zdonik, and R. Alonso, “Broadcast Disks: Data Management for Asymmetric Communications Environments,” *Proc. of ACM SIGMOD Int. Conf. on Management of Data,* pp. 199–210, 1995.

[2] R. Agrawal and R. Srikant, “Fast Algorithm for Mining Association Rules in Large Databases,” *Proc. of the 20th Int. Conf. on Very Large Data Base,* pp. 487–499, 1994.

[3] D. Barbara, “Mobile Computing and Databases – A Survey,” *IEEE Trans. on Knowledge and Data Eng.,* Vol. 11, No. 1, pp. 108–117, Jan./Feb. 1999.

[4] Y.-I. Chang and C.-N. Yang, "A Complementary Approach to Data Broadcasting in Mobile Information Systems,” *Data and Knowledge Engineering,* Vol. 40, No. pp. 181-194, 2002.

[5] Y. D. Chung and M. H. Kim, “QEM: A Scheduling Method for Wireless Broadcast Data,” *Proc. of the 6th IEEE Int. Conf. on Database Systems for Advanced Applications,* pp. 135–142, 1999.

[6] S. Hameed and N. Vaidya, “Efficient Algorithm for Scheduling Data Broadcast,“ *Wireless Networks*, Vol. 5, No. 3, pp. 183–193, 1999.

[7] V. C. S. Lee, K. W. Lam, S. Wu and E. Chan, "Broadcasting Consistent Data in Mobile Computing Environments,” *Proc. of the 7th IEEE Symp. on Real-Time Technology and Application,* pp. 123–124, 2001.

[8] G. Lee, M. S. Yeh, S. C. Lo, and Arbee L. P. Chen, “A Strategy for Efficient Access of Multiple Data Items in Mobile Environments,” *Proc. of the 3rd IEEE Int. Conf. on Mobile Data Management,* pp. 71–78, 2002.

[9] W. C. Peng and M. S. Chen, "Dynamic Generation of Data Broadcasting Programs for a Broadcast Disk Array in a Mobile Computing Environment,” *Proc. of the 9th Int. Conf. on Information Knowledge Management,* pp. 38–45, 2000.

[10] C. J. Su, L. Tassiulas, and V. Tsotras, “Broadcast Scheduling for Information Distribution,” *Wireless Networks,* Vol. 5, No. 2, pp. 137–147, March 1999.