Scheduling Dependent Items in Data Broadcasting Environment

Hao-Ping Hung[†], Jen-Wei Huang[+], Jung-Long Huang[‡] and Ming-Syan Chen[†]
[†]Graduate Institute of Communication Engineering, National Taiwan University
[+]Department of Electrical Engineering, National Taiwan University
[‡]Department of Computer Science, National Chiao-Tung University
Email: mschen@cc.ee.ntu.edu.tw,
{hphung, jwhuang}@arbor.ee.ntu.edu.tw, jlhuang@cs.nctu.edu.tw

ABSTRACT

In recent years, data broadcasting becomes a promising technique to design a mobile information system with power conservation, high scalability and high bandwidth utilization. However, most of the prior research works in data broadcasting are based on the assumption that the disseminated items are independent of one another. Since in many applications, a mobile user will be interested in more than one item simultaneously, we discuss in this paper the issue of dependency in generating a broadcast program. Algorithm PBA, standing for Placement-Based Allocation, is proposed to generate a broadcast program with high quality and low complexity in the dependent data broadcasting environment. The experimental results show that the proposed placement-based allocation for scheduling dependent items leads to better execution efficiency and solution quality than those by prior works.

Keywords

Mobile Computing, Dependent Data Broadcasting

Categories and Subject Descriptors

H.2.4 [**DATABASE MANAGEMENT**]: Systems-*Query processing*

General Terms

Algorithms, Management

1. INTRODUCTION

In recent years, the advance in wireless communication technology changes the paradigm in which information is exchanged. Users can ubiquitously access the remote data via mobile appliances. Due to the characteristics of asymmetric communication, frequent disconnection and power limitation, providing scalable and preferable services in the mobile computing environment becomes an emerging topic. In order to maintain the information on the air, researchers have encountered and are endeavoring to overcome the challenges in various fields including data dissemination [1][2][8], wireless cache [15], location-dependent query processing [14][16], mobility management [3][5], and so on.

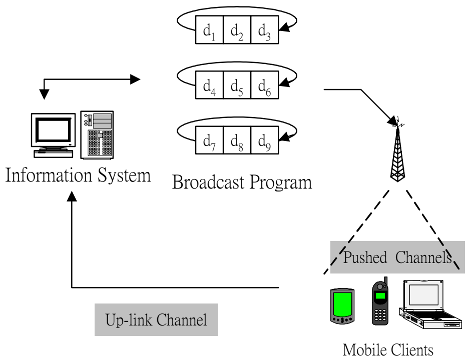


Figure 1: The architecture of broadcast-based information system

Among various aspects in mobile computing technologies, data broadcasting has received the most attention in prior research. The broadcasting technique is a scalable way to disseminate the data items from an information system to the interested mobile clients. Figure 1 shows the architecture of a broadcast-based information system. The physical bandwidth is partitioned into several logical channels. The server generates a *broadcast program* by collecting the access patterns of mobile users, and delivers data items *periodically*. To retrieve the data, Mobile users should wait until items of interest appear in channel. Such a technique is initiated in [1] and extended by Hsu [7], Yee [17], and Hung [10], etc. In [1], the concept of broadcast disks is first proposed to provide scalable information services. To schedule the data items more effectively in broadcast channels, the authors in [7] focus on generating a near optimal broadcast program. In [17], the optimal broadcast program can be generated based on the concept of *dynamic programming*. As for [10], the heterogeneous data broadcasting environment is considered.

In this paper, we focus on scheduling the *dependent* data items into multiple broadcast channels. In conventional broadcasting environment, the scheduling policy is based on the assumption that each data item is independent of one another. However, in many applications, a mobile user may be interested in multiple items simultaneously. For example, a mobile user interested in the stock information of one company may also request for the information of other relevant companies. Therefore, the data items in a query (i.e., the request sent from clients to the server for information service) should be considered *dependent* on one another. To generate a broadcast program in the *dependent data broadcasting* environment, the server should allocate the data items into single or multiple channels by collecting the query profile (i.e., the table storing the inter-relationship between each query and its corresponding items) and the access patterns. To retrieve the items of interest, a mobile user should switch from one channel to another until all the items contained in his/her query are downloaded. Note that the requested items in a query can be retrieved in an arbitrary order. Moreover, if more than one queried item appears in different channels simultaneously, only one of them is allowed to be downloaded. The remaining items should be retrieved in succeeding broadcast cycles. In the dependent data broadcasting environment, it is possible for one item to be contained by more than one query. Given a specific item, it may be dependent on one item in one query and on another item in another query. Therefore, to find an optimal scheduling policy, it costs the complexity of $O(N!)$ to search all possible solutions thoroughly, where N denotes the number of items in the database. The difficulty of the scheduling policy was addressed in either single channel [11] or multi-channel [9] environment. To provide better scheduling policies, a significant amount of research effort has been elaborated in this aspect [4][9][11][12]. In [4], a broadcast scheduling method QEM, standing for Query Expansion Method, is proposed to minimize the average access time. In algorithm QEM, the items in popular queries have better chances to be scheduled. Algorithm QEM can be viewed as a *greedy* algorithm since it will choose the most profitable position to *append* the items in a query. In [11], the authors propose an extended scheduling method by loosening the restrictions in [4]. In [12], since a broadcast program is regarded as a *permutation* of the items, the problem of broadcasting dependent data can thus be formulated as finding a *permutation* with the minimum cost. It is noted that the prior works in [4][11][12] only consider the *single* channel environment. As for the multi-channel environment, an effective approach GA, which is based on the concept of *genetic algorithm* [6], is proposed in [9].

Although GA has been proved to be a versatile approach for scheduling policy in dependent data broadcasting environment, there are still many situations in which the genetic algorithm may result in poor performance such as slow convergence and false optimum. Moreover, the performance of the genetic algorithm will be affected by the initial setting. Since the initial setting is generated randomly, there is no guarantee for the genetic algorithm to result in good quality. To avoid such drawbacks induced by stochastic search, we propose in this paper a deterministic algorithm named PBA, standing for Placement-Based Allocation, to schedule the items into different channels. Algorithm PBA generates a broadcast program from the viewpoint of the *placement*, which is defined as the location for a data item to appear in a channel. Since each query may contain multiple items in dependent data broadcasting environment, the scheduling policy of PBA is to avoid the occurrence of the *conflicting items*, defined as the situation that multiple items in a query are located in the same placement. Moreover, algorithm PBA allows the query with higher access probability to have higher priority to schedule its items. Such provision can effectively reduce the impact of the *conflicting items*. Since algorithm PBA scans the database only once, it can generate the broadcast program very efficiently with high quality.

The rest of this paper is outlined as follows. In Section 2, notation and definitions are given. In Section 3, we will describe the proposed placement-based allocation algorithm PBA. The experimental results will be shown in Section 4, and finally, this paper concludes with Section 5.

Table 1: Description of the symbols

Parameters	Descriptions
D	The collection of broadcast items
Q	The collection of queries issued by cilents
K	Number of channels
L	The length of the broadcast cycle
q_i	The i-th query in Q
d_j	The j-th item in D
s	Size of each data item
B	Bandwidth of each channel

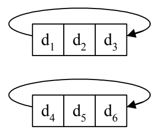


Figure 2: An example of the broadcast program

2. NOTATION AND DEFINITIONS

Table 1 summarizes the symbols and corresponding descriptions. Suppose that database D contains $|D|$ items. Let Q with size $|Q|$ represent the *query profile*, i.e., the collection of queries that mobile users may issue. A query $q_i \in Q$ is regarded as a request for single or multiple items. Given the access patterns of queries, an information system will generate a broadcast program P, which distributes the items in D into K channels evenly. The length of the broadcast cycle, denoted by L, can thus be formulated as $L = \lceil \frac{|D|}{K} \rceil$. Note that L can be viewed as the number of the *placements* in P.

Definition 1: Given a specific query q_i, $cycle_i$ is defined as the number of complete broadcast cycles that a mobile user should spend downloading the items in q_i.

Definition 2: Given a specific query q_i, the *refined query* q_i' is defined as the set of remaining items that the interested client needs to download after waiting for $cycle_i$ cycles.

Example 1: Consider the broadcast program in Figure 2. Suppose that $q_i = \{d_1, d_2, d_4\}$. Since d_1 and d_4 are located in the same placement, the client issuing the query should wait for one complete broadcast cycle to download $\{d_1, d_2\}$ (respectively, $\{d_4, d_2\}$), and then downloads the remaining item d_4 (respectively, d_1). Therefore, $cycle_i = 1$ and the refined query $q_i' = \{d_4\}$ or $q_i' = \{d_1\}$.

Definition 3: Given a refined query q_i', the *start-up* time, denoted by $T_{Startup}(q_i')$, is defined as the duration of time since a mobile user issues query q_i until the $first$ item of interest appears in one of the channels. The derivation of $T_{Startup}(q_i')$ can be found in [9].

Definition 4: Given a refined query q_i', the *retrieval* time, denoted by $T_{Retr.}(q_i')$, is defined as the duration of time since the first item contained in query q_i' appears in one of the channels until all the items in q_i' are downloaded. The derivation of $T_{Retr.}(q_i')$ can be found in [9].

Definition 5: The access time of a refined query q_i', denoted by $T_{Access}(q_i')$, is defined as the sum of the *start-up* time and the *retrieval* time. i.e.,

$$T_{Access}(q_i') = T_{Startup}(q_i') + T_{Retr.}(q_i').$$

Definition 6: Let s and B represent the size of each item and the bandwidth of each channel, respectively. The access time of the query q_i, denoted by $T_{Access}(q_i)$, is defined as the duration from the time when a query is issued to the time when all items of interest are downloaded. It is noted that the access time of a query can be decomposed into the access time of the $refined\ query$ and the duration of $cycle_i$ broadcast cycles. The access time of q_i can be formulated as

$$T_{Access}(q_i) = T_{Access}(q_i') + cycle_i \times L \times \frac{s}{B}.$$

Definition 7: The average access time of the query profile Q, denoted by $T_{Access}(Q)$, is defined as the average access time of the clients issuing a query. i.e.,

$$T_{Access}(Q) = \sum_{i=1}^{|Q|} [T_{Access}(q_i) \times Pr(q_i)],$$

where $Pr(q_i)$ represents the access probability of q_i.

3. PLACEMENT-BASED ALLOCATION ALGORITHM

The design of algorithm PBA focuses on generating a high broadcast program in an efficient way so as to suit the variation of the access patterns in the dynamic environment. In order to achieve efficient allocation, we deal with the problem by investigating the analytical model of dependent data broadcasting environment. From the observation of Definition 6, two critical terms contribute $T_{Access}(q_i)$: the refined query q_i' and the number of complete broadcast cycles $cycle_i$. The intuition of algorithm PBA is that the items should be allocated in such a way that $cycle_i$ is minimized. To facilitate the following descriptions, the *conflicting items* are defined as follows.

Definition 8: The *conflicting items* of a query are defined as the set of *multiple* items, which belong to the same query, located in the same placement.

Example 2: Following Example 1, the *conflicting items* of q_i are $\{d_1, d_4\}$ since d_1 and d_4 are both located in the first placement of the broadcast program.

In order to minimize $T_{Access}(Q)$, algorithm PBA is designed from the viewpoint of *placement* of the broadcast program. Note that mobile users should spend more cycles downloading the *conflicting items*. Moreover, a broadcast program with cycle length L can be viewed as a collection which contains L item sets. The problem of generating a broadcast program can be reformulated as the problem of inserting the items of each query into L groups such that the occurrence of *conflicting items* is avoided. Denote the broadcast program P as $P = \{D_1, D_2, ..., D_L\}$, where $D_1, ..., D_L$ represent the disjoint subsets of D and each D_i contains no more than K items. The algorithmic form of algorithm PBA is outlined as follows.

```
Algorithm PBA
Input: the query profile $Q$,
       the broadcast database $D$,
       the number of channels $K$
Output: the broadcast program $P$
Begin
1. Sort elements in $Q$ according to the access probability
   in descending order.
2. Let $L = \lceil \frac{|D|}{K} \rceil$. Create an empty broadcast program
   which contains $L$ item sets,
   i.e., $P = \{D_i, 1 \le i \le L\}$ where $D_i = \{\varnothing\}$
3. for each $q_i$ in $Q$
4.    Unmark all item sets
5.    Find $q_i^+$ and $q_i^-$ which store the scheduled and
      unscheduled items in $q_i$, respectively
6.    Mark the item sets which contain $K$ items and
      the item sets which contain the items in $q_i^+$
7.    if all item sets are marked
8.       $curr = \arg_i \min\{|D_i|\}$ for all $D_i$, $|D_i| < K$
9.    else
10.      $curr = \arg_i \min\{|D_i|\}$ for all unmarked $D_i$
11.   for each item $d_j$ in $q_i^-$
12.      Insert $d_j$ into $D_{curr}$
13.      Mark $D_{curr}$
14.      if all item sets are marked
15.         Select $next$ from $\{D_i, |D_i| < K\}$
            such that $DIST(curr, next)$ is minimized
16.      else
17.         Select $next$ from the unmarked item sets
            such that $DIST(curr, next)$ is minimized
18.      Set $curr = next$
19.return $P$
End
```

```
Function $DIST(i, j)$
Input: the placements $i, j$,
       the length of the broadcast cycle $L$
Output: the distance from placement $i$ to placement $j$
Begin
1. if $j > i$
2.    return $j - i$
3. else
4.    return $j + L - i$
End
```

To allocate the data items, algorithm PBA will first sort the queries in Q according to the access probability in descending order. A query with higher access probability deserves higher priority to be scheduled. After that, a broadcast program P, which contains L empty item sets, is created. It is noted that the item set D_i corresponds to the items allocated in the i-th placement. When each query q_i is processed, the items which have been allocated should not be scheduled again. To prevent the items from being allo-

Table 2: Query profile of the example

query	prob.	items
q_1	0.310	$d_9, d_{12}, d_{18}, d_{19}$
q_2	0.178	$d_{10}, d_{11}, d_{13}, d_{17}, d_{18}$
q_3	0.128	$d_3, d_{14}, d_{16}, d_{17}, d_{18}$
q_4	0.102	$d_2, d_5, d_6, d_{11}, d_{14}, d_{15}$
q_5	0.085	$d_2, d_{13}, d_{17}, d_{20}$
q_6	0.074	$d_1, d_{11}, d_{17}, d_{20}$
q_7	0.065	d_3, d_4, d_{15}, d_{19}
q_8	0.058	d_7, d_8, d_{14}, d_{18}

cated more than once, we divide q_i into two disjoint subsets, q_i^+ and q_i^-, which store the scheduled and unscheduled items, respectively. Next, some of the item sets in P will be marked. The marking principles are as follows: (1), the item sets which contain *exactly* K items should be marked, and (2), the item sets which contain the items in q_i^+ should be marked. Notice that the *marked* item sets indicate the placements where items in q_i^- should not be allocated. The reason of the marking procedure is that inserting the items into a marked item set will lead to either the illegitimacy of the broadcast program or an occurrence of *conflicting items*. To allocate an item d_j in q_i^-, algorithm PBA will determine a suitable placement, denoted by $curr$, for d_j to insert. After inserting d_j into the item set D_{curr} which corresponds to the placement $curr$, D_{curr} will also be marked. To allocate the next item in q_i^-, the next placement, denoted by $next$, will be determined according to the *distance function* $DIST(i, j)$. Specifically, the function $DIST(i, j)$ returns the distance from the i-th placement to the j-th placement in P by taking the effect of periodicity into account. e.g., in Figure 2, $DIST(1, 3) = 2$ whereas $DIST(3, 1) = 1$. Algorithm PBA will terminate its execution when all the items in D are allocated in P. From the perspective of algorithm PBA, the items in *popular* queries, i.e., the queries with high access probability, have high priority to be scheduled. When the items of an unpopular query are going to be placed, there is much less flexibility. Although it is inevitable for some unpopular queries to encounter the occurrences of *conflicting items*, such discrimination can effectively reduce the impact of the *conflicting items*. Note that algorithm PBA only scans the items in D once, it is very efficient and scalable for algorithm PBA to generate a broadcast program especially when the access patterns and the contents of the database change dynamically.

Finally, we use a running example to illustrate algorithm PBA.

Example 3: An example of executing algorithm PBA is presented in Figure 3. According to the query profile in Table 2, eight queries are first sorted according to their access probabilities. Assuming that 20 items will be disseminated via four channels, the broadcast program P creates five empty item sets labeled as $1 \sim 5$ in the beginning. In Figure 3(a), items contained in q_1 and q_2 are scheduled into the broadcast program. In order to distinguish the items in different queries, we use lighter and darker shadows to identify the items in q_1 and q_2, respectively. To put the items in q_1 into P, since this query has the highest priority, all of the items in q_1 will be put in the placement $1 \sim 4$. The items in q_2 will be processed following the scheduling

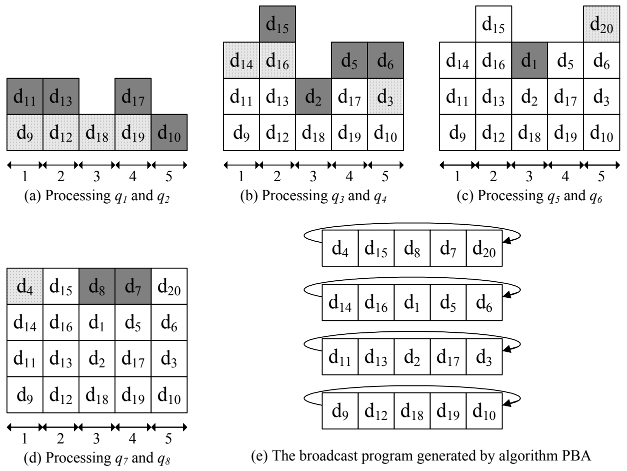


Figure 3: The example of executing algorithm PBA

result of q_1. Items in q_2 will be partitioned into two parts depending on whether an item already exists in P or not. Since d_{18} already exists in P, q_2^+ contains $\{d_{18}\}$, whereas q_2^- contains $\{d_{10}, d_{11}, d_{13}, d_{17}\}$. To determine the placement where the first item in q_2^-, i.e., d_{10}, should be allocated, algorithm PBA selects placement 5, since the corresponding D_5 contains the fewest items among all unmarked item sets. After that, the other items in q_2^-, i.e., $\{d_{11}, d_{13}, d_{17}\}$, will be allocated in placements 1, 2, 4, respectively. Note that placement 3 is skipped since it is marked to avoid the conflict to d_{18}. Figure 3(b) and Figure 3(c) depict the execution when algorithm PBA processes the items in $q_3 \sim q_6$. When an item set is filled with 4 items, it will also be marked to avoid the illegitimacy of P. As the the number of marked placements increases, there will be much less flexibility to schedule the items with lower priority. As shown in Figure 3(d), when the item d_8 in q_8^- is scheduled, it is inevitable to face the situation that all placements are marked. Under this circumstance, algorithm PBA will put the item into the placement in which the number of the items is less than 4. After all the items are scheduled, algorithm PBA will return P. In Figure 3(e), the data items are broadcast according to the scheduling result of P.

4. EXPERIMENTAL RESULTS

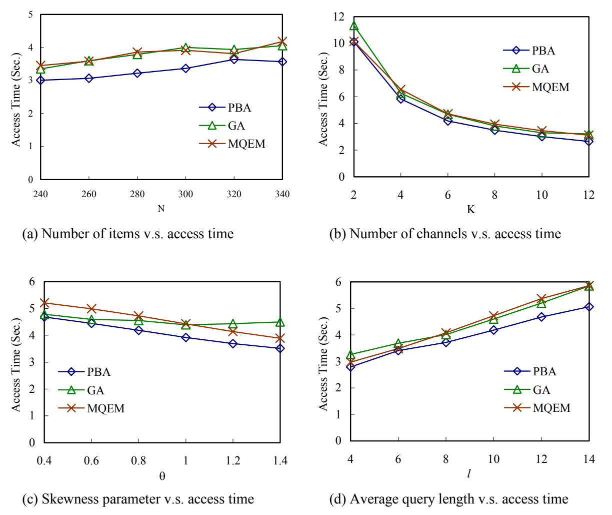
In this section, we conduct several experiments to investigate the performances of algorithm PBA. During the experiments, several well-known approaches in prior works [9][11] are also investigated empirically. We will describe the simulation environment in Section 4.1. In Section 4.2, we will discuss the effect of several parameters including the number of items, the number of channels, the skewness and the average query length. In the last experiment, the efficiency analysis will be given in Section 4.3.

4.1 Simulation Environment

In Table 3, several simulating parameters are listed. The access probability of each query is generated by Zipf distribution $Pr(q_i) = (\frac{1}{i})^\theta / \sum_{j=1}^{|D|} (\frac{1}{j})^\theta$, where θ is the skewness parameter and $1 \leq i \leq |D|$. The query profile is generated based on the approach mentioned in [13], in which

Table 3: Values of the simulation parameters

Parameters	Values
Number of the items (N)	$240 \sim 340$
Number of the queries	100
Number of the channels (K)	$2 \sim 12$
Average length of a query (l)	$4 \sim 14$
Skewness parameter (θ)	$0.4 \sim 1.4$
Bandwidth of each broadcast channel	80K bytes/sec
Size of each data item	8K bytes



(a) Number of items v.s. access time

(b) Number of channels v.s. access time

(c) Skewness parameter v.s. access time

(d) Average query length v.s. access time

Figure 4: The effectiveness analysis of algorithm PBA

the inter-relationship among the items can be modeled as a dependency graph. From the experimental results, we observe that the number of the queries is viewed as a minor factor affecting the performance. Therefore, in the following experiments, the number of queries is fixed to be 100. We compare the performance of PBA to algorithm GA [9]. Moreover, we also extend the concept of Modified Query Expansion Method in [11] from single channel broadcasting to the multi-channel environment and denote it as MQEM.

4.2 Effectiveness Analysis

In the first experiment, we discuss the effect of the number of items, denoted by N. The amount of $T_{Access}(Q)$ is used to assess the quality of broadcast programs generated by different algorithms. As shown in Figure 4(a), algorithm PBA, which generates broadcast programs by avoiding the occurrences of *conflicting items*, outperforms the conventional genetic algorithm GA. On the other hand, algorithm MQEM only generates broadcast programs with fair quality similar to GA. The reason is that the ignorance of the multi-channel effect will degrade the performance of algorithm MQEM. We also observe that the access time of each scheme tends to increase as the value of N increases. This phenomenon can be explained as follows. When more items need to be broadcast, each channel requires a longer broadcast cycle to deliver the items. The increase of broadcast cycle will raise the average waiting time of mobile users.

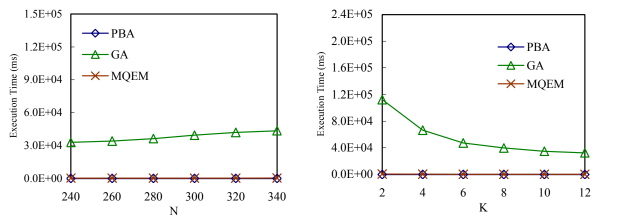
We next investigate the effect of the number of broadcast channels. Unlike the previous experiment, in Figure 4(b), the average access time of all schemes decreases as the value of K increases. The reason is that under the condition of the fixed N, K is in inverse proportion to L, i.e., $L = \lceil \frac{N}{K} \rceil$. A larger value of K will thus imply a shorter broadcast cycle. This result also agrees with our intuition since the increase of the network bandwidth causes the access time to decrease. Figure 4(b) also shows that algorithm GA incurs the worse performance for smaller K. This phenomenon is due to the inverse proportion between K and L. The situation of larger L will expand the choices of the placements when the items are allocated. Therefore, for smaller value of K (i.e., larger value of L), the advantage of PBA over GA will be more prominent. This figure also shows that algorithm MQEM performs well when K is small. This can also be explained by the reason that the multi-channel effect becomes implicit under such circumstances.

The effect of varying the skewness parameter θ can be investigated in Figure 4(c). In this experiment, the proposed algorithm PBA still outperforms algorithm GA and algorithm MQEM. The major observation in this figure is that the performance of PBA is very close to that of GA for smaller θ. With the increase of the skewness parameter, the discrepancy between PBA and GA becomes more significant. This phenomenon can be explained from the characteristics of algorithm PBA. As described in Section 3, the query with higher access probability will have higher priority to allocate the items. Since the access probabilities of the queries are close to one another for smaller θ, the priority of the queries will thus be indefinite. An extreme case, $\theta = 0$, indicates that the access probabilities of all queries are equal. The indefinite priority will degrade the performance of algorithm PBA. Note that algorithm MQEM quantifies the priority according to the access probability as well. It also results in the poor performance for lower skewness.

In addition, we discuss the effect of average query length, i.e., the average number of items contained in a query. As shown in Figure 4(d), the average access time of each scheme increases as the average query length gets larger. The reason is as follows. When more items are contained in a query, it will be more likely for *conflicting items* to occur. For an extreme example, if a query contains more than L items, it is inevitable that at least two items in the query will be allocated in the same placement regardless of its scheduling priority. As described in Section 3, mobile users should spend more time downloading the *conflicting items*. The longer average query length will thus induce the larger amount of access time.

4.3 Efficiency Analysis

In the final experiment, we discuss the efficiency issue of all algorithms. Since we implement these approaches by Java language and execute the simulators on the same system platform, the execution time of simulators will reflect the efficiency. We use millisecond as the unit of execution time. Note that the parameters θ and l do not affect the complexity in this experiment. We thus only consider the parameters N and K here. Figure 5(a) shows the execution time of each approach as the number of broadcast items increases, while Figure 5(b) depicts the execution time with K varied. We can observe that algorithm PBA and algorithm MQEM take much less execution time compared to algo-



(a) Number of items v.s. execution time (b)Number of channels v.s. execution time

Figure 5: The efficiency analysis of algorithm PBA

rithms GA. Moreover, since the increase of N will enlarge the searching space, the execution time of GA will increase drastically. On the other hand, the decrease of K will raise the length of the broadcast cycle. Since the increase of L will also enlarge the searching space, we observe that in Figure 5(b), it requires longer execution time for algorithm GA for smaller value of K.

5. CONCLUSION

In this paper, we study the scheduling algorithm in dependent data broadcasting environment. Algorithm PBA, standing for Placement Based Allocation, is proposed from the viewpoint of *placement*. According to the experimental results, PBA can achieve better performance in both effectiveness and efficiency compared to the prior works. Therefore, the proposed algorithm PBA will be very suitable and practical to generate the broadcast program in the dependent data broadcasting environment.

6. REFERENCES

[1] S. Acharya, R. Alonso, M. J. Franklin, and S. B. Zdonik. Broadcast Disks: Data Management for Asymmetric Communications Environments. In *Proceedings of the 1995 ACM International Conference on Management of Data*, pages 199–210, May 1995.

[2] D. Aksoy and M. Franklin. RxW: A Scheduling Approach for Large-Scale On-demand Data Broadcast. *IEEE/ACM Transactions on Networking*, 7(6):846–860, 1999.

[3] I. R. Chen. Quantitative Analysis of a Hybrid Replication with Forwarding Strategy for Efficient and Uniform Location Management in Mobile Wireless Networks. *IEEE Tran. on Mobile Computing*, 2(1), 2003.

[4] Y. D. Chung and M. H. Kim. Effective Data Placement for Wireless Broadcast. *Distributed and Parallel Databases*, 9(2), 2001.

[5] Y. Fang. Movement-Based Mobility Management and Trade Off Analysis for Wireless Mobile Networks. *IEEE Tran. on Computers*, 52(6), 2003.

[6] D. E. Goldberg. Genetic Algorithm in Search, Optimization and Machine Learning. *Addison-Wesley Publishing*, 1989.

[7] C.-H. Hsu, G. Lee, and A. L. P. Chen. A near optimal algorithm for generating broadcast programs on multiple channels. In *Proceedings of the 10th ACM International Conference on Information and Knowledge Management*, pages 303–309, November 2001.

[8] C.-L. Hu and M.-S. Chen. Adaptive Multi-Channel Data Dissemination: Support of Dynamic Traffic Awareness and Push-Pull Time Balance. *Trans. on Vehicular Technology*, 54(2), 2005.

[9] J.-L. Huang and M.-S. Chen. Dependent Data Broadcasting for Unordered Queries in a Multiple Channel Mobile Environment. *IEEE Trans. on Knowledge and Data Engineering*, 16(6), Jun. 2004.

[10] H.-P. Hung and M.-S. Chen. On Exploring Channel Allocation in the Diverse Data Broadcasting Environment. In *IEEE ICDCS*, 2005.

[11] G. Lee and S. C. Lo. Broadcast Data Allocation for Efficient Access of Multiple Data Items in Mobile Environments. *Mobile Networks and Applications*, 8, 2003.

[12] F. Martinez, J. Gonzalez, and I. Stojmenovic. A Parallel Hill Climbing Algorithm for Pushing Dependent Data in Clients-Providers-Servers Systems. In *Proceedings of Computer and Communications (ISCC'02)*, 2002.

[13] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos. Effective Prediction of Web-User Accesses: A Data Mining Approach. In *Proc. of WEBKDD Workshop*, Auguest 2001.

[14] Q. Ren. Using Semantic Caching to Manage Location Dependent Data in Mobile Computing. In *Procedings of ACM MOBICOM*, 2000.

[15] J. Xu, Q. Hu, W.-C. Lee, and D. Lee. Performance Evaluation of an Optimal Cache Replacement Policy for Wireless Data Dissemination. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):125–139, Jan. 2004.

[16] J. Xu, X. Tang, and D. Lee. Performance Analysis of Location-dependent Cache Invalidation Schemes for Mobile Environments. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):474–488, Mar/Apr 2003.

[17] W. G. Yee, S. B. Navathe, E. Omiecinski, and C. Jermaine. Efficient data allocation over multiple channels at broadcast servers. *IEEE Trans. Computers*, 51(10), 2002.