Broadcast Program Generation for Unordered Queries with Data Replication

Jiun-Long Huang and Ming-Syan Chen
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, ROC
E-mail: jlhuang@arbor.ee.ntu.edu.tw, mschen@cc.ee.ntu.edu.tw

ABSTRACT

We study in this paper the problem of broadcasting dependent data for unordered queries. However, most prior studies on dependent data broadcasting are limited to the premise of no data replication. Different from other prior studies, we investigate the effect of data replication in this paper. Specifically, we first derive several theoretical properties for the average access time by analyzing the model of dependent data broadcasting. On the basis of the theoretical results, we develop a genetic algorithm to generate broadcast programs with replication. In order to compare the performance of the proposed algorithm and the prior studies, several experiments are conducted. Our experimental results show that with the analytical results derived, the theoretical results derived are able to guide the search of the genetic algorithm very effectively, and lead to solution broadcast programs of higher quality than those of the prior studies.

Keywords

Data broadcast, mobile information system, mobile computing, genetic algorithm

1. INTRODUCTION

Most works in data broadcasting were under the premise that each user requests only one data item at a time and the requests for all data items are independent. However, in many real applications, there exists dependency among data items, and a mobile user may submit a query to retrieve *multiple* data items. Explicitly, queries of multiple, dependent data can be categorized into the following two types according to the constraint of the sequence of these data items:

Ordered queries In an ordered query, the required data items should be retrieved in a predetermined order.

SAC 2003, Melbourne, Florida, USA

Unordered queries Similar as an ordered query, an unordered query could be one issued by a mobile user for requesting multiple data items simultaneously. However, unlike an ordered queries, these required data items may be retrieved in any order.

In both types of queries, data allocation algorithms assuming independent requests are not able to effectively optimize the performance of the broadcast programs. This phenomenon attracts a series of studies on solving the problem of dependent data broadcast. Note that for each ordered query, the required data items of this query should be retrieved according to a predetermined order, and there will be exactly one retrieval order. In contrast, the number of retrieval orders of an unordered query Q_i is $|Q_i|!$ where $|Q_i|$ is the number of required data items of Q_i. This feature makes the broadcasting of dependent data for unordered queries be more difficult than that for ordered queries.

It is noted that the most prior studies of dependent data broadcast do not consider data replication. That is, each data item appears *exactly once* in the broadcast programs as shown in the example in Figure 1b. A broadcast program without replication is also called a *flat* broadcast program. As shown in the prior studies on *independent* data broadcast [1], employing data replication in broadcast program generation is able to optimize the broadcast programs more effectively than flat broadcast programs especially when the access probability for each data item is skewed. Let $D_i(j)$ represent the j-th copy of data item D_i. As shown in Figure 1c, one can replicate hot data items (i.e., items with high access probabilities) into several copies in the broadcast program to further reduce the average access time. However, none of the above studies on dependent data broadcasting considered data replication which is practically important.

Consequently, we address the problem of the broadcast program generation for dependent data with unordered queries in this paper. Unlike [2][6], data replication is employed in our study. Specifically, several special cases of the problem of broadcasting dependent data are shown to be NP-hard [2]. In view of this, we shall employ the Genetic Algorithm [3] (abbreviated as GA) in this paper to address the problem of broadcasting dependent data for unordered queries with data replication. Basically, GAs are iterative procedures that search the problem solutions by an evolutionary process based on natural selection. GA maintains a population of individual candidate solutions to specific problems. An individual candidate solution can be represented as a list called a *chromosome*. In GA, a *fitness function* has to be

Item	Size	Item	Size	Item	Size
D_1	6	D_3	5	D_5	5
D_2	8	D_4	6	D_6	7

(a) The sizes of all data items

D_5	D_3	D_1	D_6	D_4	D_2

(b) The broadcast program without data replication

$D_6(2)$	$D_1(1)$	$D_2(1)$	$D_1(2)$	$D_3(1)$	$D_5(1)$	$D_6(1)$	$D_4(1)$	$D_3(2)$

(c) The broadcast program with data replication

Figure 1: An example of a broadcast program with data replication

Query(Q_i)	$Pr(Q_i)$
$Q_1 = \{D_1, D_2, D_3\}$	50%
$Q_2 = \{D_1, D_3, D_4\}$	20%
$Q_3 = \{D_4, D_6, D_1\}$	20%
$Q_4 = \{D_1, D_2, D_5\}$	5%
$Q_5 = \{D_5, D_3, D_4, D_6\}$	5%

Figure 2: An example query profile

designed to evaluate the fitness of each chromosome,and the design of the fitness function is key to the effectiveness of the GA algorithms.

Explicitly, in this paper we first model the problem of broadcast program generation for unordered queries with replication. By analyzing the model of dependent data broadcasting, we derive several theoretical properties for the average access time. In light of the theoretical results, we then formulate the fitness function for one GA to generate broadcast programs with replication. Sensitivity analysis on several parameters is conducted. Our experimental results show that with the analytical results derived, the theoretical results derived are able to guide the search of the genetic algorithm very effectively, and lead to solution broadcast programs of higher quality than those of the prior studies. In addition, the experimental results also show that the data replication technique employed can lead to more efficient use of network bandwidth. To the best of our knowledge, there is no prior work on the broadcast program generation for unordered queries with data replication. This fact distinguishes this paper from others.

The rest of this paper is organized as follows. Section 2 presents the preliminaries of this study. Analytical models of the problem of broadcasting dependent data with unordered queries are derived in Section 3. In light of the analysis in Section 3, we devise a GA-based algorithm in Section 4. Performance evaluation on varies parameters is conducted in Section 5. Finally, Section 6 concludes this paper.

2. PROBLEM FORMULATION

Same as in [1], it is assumed that the database D contains $|D|$ data items, $D_1, D_2, \cdots, D_{|D|}$ and each data item is read-only. The size of D_i is assumed to be $size(D_i)$. Thus, the size of the whole database is $size(D) = \sum_{i=1}^{|D|} size(D_i)$. From the users' perspective, a query is an indivisible request of single or multiple data items as defined below.

Definition 1 An *unordered* query $Q_i = \{D_{q^i(1)}, D_{q^i(2)}, \cdots, D_{q^i(|Q_i|)}\}$ is a non-empty subset of all data items where $|Q_i|$ represents the number of required data items in Q_i. Note that $1 \leq q^i(j) \leq |D|$ for all j where $1 \leq j \leq |Q_i|$, and $q^i(j) = k$ means that the j-th accessed data item in Q_i is D_k.

The query profile is the aggregation of the access behavior of all users. Formally, we have the following definition.

Definition 2 A query profile Q consists of a set of $\langle Q_i, Pr(Q_i)\rangle$ pairs where $|Q|$ indicates the number of queries in Q. $Pr(Q_i)$ represents the probability that a query Q_i is issued by users. It is noted that $\sum_{i=1}^{|Q|} Pr(Q_i) = 1$.

The problem of broadcast program generation with replication can be divided into two subproblems: (1) determining the number of replicas needed for each data item and (2) determining the placement of these replicas into the broadcast program. The first subproblem indicates that the system should determine how many copies for each data item to be placed in the broadcast program. According to the property shown in [4], the total average access time will be minimized if the copies of each data item are equally spaced and for any two data items D_i and D_j,

$$\frac{n(D_i)}{n(D_j)} = \frac{\sqrt{\frac{Pr(D_i)}{size(D_i)}}}{\sqrt{\frac{Pr(D_j)}{size(D_j)}}}. \quad (1)$$

where $n(D_i)$ is the number of replicas of D_i and $Pr(D_i)$ is the access frequency of D_i. $Pr(D_i)$ can be obtained by the following equation:

$$Pr(D_i) = \sum_{j=1}^{|Q|} \Big(Pr(Q_j) \times \text{the number of occurrences of } D_i \text{ in } Q_j \Big).$$

Let L be the length of broadcast program with replication as determined in accordance with system capacity. Note that $L \geq size(D)$ since all data items should be broadcast at least once. With the same reason, at first each data item appears exactly once in the broadcast program. There will be a space with size $L - size(D)$ left. Denote the number of extra copies of D_i appearing in the rest of the broadcast program as $n'(D_i)$. Then we have

$$\sum_{i=1}^{|D|} n'(D_i) \times size(D_i) = L - size(D).$$

Let the relationship of $n'(D_i)$, $i = 1, 2, \cdots, |D|$ follow Equation (1). Thus, the above equation can be rewritten as

$$n'(D_1) \times \sum_{i=1}^{|D|} \left[\frac{n(D_i)}{n(D_1)} \times size(D_i) \right] = L - size(D).$$

Since only one unknown variable is in the above equation, $n'(D_1)$ can be solved. All other $n'(D_i)$s are obtained by Equation (1), and then, $n(D_i)$ is determined as $\lceil n'(D_i) \rceil + 1$.

After determining $n(D_i)$ for each data item D_i, we revise the original database by considering the number of replicas for each data item as D^* which is defined as follows,

$$D^* = \bigcup_{i=1}^{|D|} \left(\bigcup_{j=1}^{n(D_i)} \left\{ D_i(j) \right\} \right),$$

where $D_i(j)$ indicates the j-th copy of data item D_i. The size of the revised database D^* (denoted as $size(D^*)$) is as follows,

$$size(D^*) = \sum_{i=1}^{|D|} n(D_i) \times size(D_i) = L.$$

After determining the number of replicas for each data item, the broadcast program with replication can be stated as follows.

Definition 3 A broadcast program P *with* replication is a *placement* of all data items in D^* into a list with length L, where L is a predetermined number and $L \geq size(D)$. In addition, each data item D_i will appear $n(D_i)$ times in the broadcast program P.

To facilitate the further discussion, we utilize the function $offset(i, j)$ to represent the offset of the j-th copy of D_i in the broadcast program. $offset(i, j)$ is equal to the summation of the sizes of all data items with smaller broadcast order than the j-th copy of D_i.

We take access time as the measurement for the quality of broadcast programs. As a result, given the number of broadcast channels, the database D and a query profile Q, the problem of broadcast program generation with replication is to determine a broadcast program P with replication which minimizes the average access time of the query profile Q. Denote, respectively, the average access time of a query Q_i as $T_{Access}(Q_i)$ and the average access time of a query profile Q as $T_{Access}(Q)$. The average access time of the query profile Q can be formulated as the following equation,

$$T_{Access}(Q) = \sum_{i=1}^{|Q|} \Big[T_{Access}(Q_i) \times Pr(Q_i) \Big]. \qquad (2)$$

3. ANALYTICAL MODELS

To facilitate the derivation of the average access time, we first decompose the access time of an query Q_i into two parts:

1. *Startup time*: When a mobile user submits a query Q_i, the mobile device should wait until the system starts to broadcast any required data item of Q_i. This time interval is called startup time.

2. *Retrieval time*: Retrieval time is defined as the time intervals between the moment that the mobile device starts to read data items from broadcast channels and that the mobile device finishes Q_i.

Obviously, the access time of an query is the summation of the startup time and retrieval time. Denote the bandwidth of each broadcast channel as B. Then we have the following example.

Without loss of generality, we assume that the user submits Q_i in the m-th broadcast cycle and the time interval between the start time of the m-broadcast cycle and the time that the user submits Q_i (i.e., s_0 in Figure 3) is a uniform distribution over $(0, L)$. To facilitate the following discussion, we define the *candidates* of the first retrieved data items of Q_i as follows.

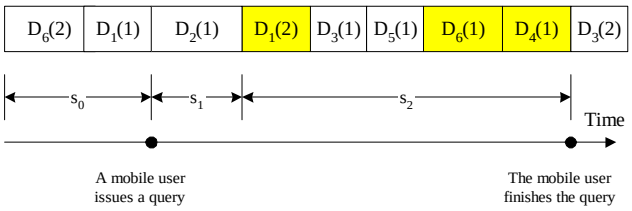

Figure 3: An example scenario of a query

Definition 4 The *candidates* of the first retrieved data items of Q_i , $Cand = \{Cand(1), Cand(2), \cdots\}$, is an *ordered* set of all copies of date items in Q_i. $Cand$ is sorted according to the offsets of all elements in an ascending order. In addition, the number of the candidates of Q_i is equal to $\sum_{i=1}^{|Q_i|} n(D_{q_j^i})$.

To simplify the further derivation, we define a series $a(j)$, $j = 1, 2, \cdots, b$, to represent the offsets of all data item in $Cand$ (i.e., $a(j)$=the offset of the data item $Cand(j)$), where b represents the number of candidates (i.e., $b = |Cand|$).

Denote the average startup time of Q_i as $T_{Startup}(Q_i)$. We have

$$\begin{aligned} & T_{Startup}(Q_i) \\ = \; & \frac{1}{L \times B} \times \left\{ \sum_{i=1}^{b-1} \frac{\Big[a(i+1) - a(i)\Big]^2}{2} + \frac{\Big[L - a(b) + a(1)\Big]^2}{2} \right\} \end{aligned} \qquad (3)$$

In order to derive the average retrieval time of Q_i, we first let $p(j)$ represent the probability that the user retrieves "the j-th candidate (i.e., $Cand(j)$) of the first retrieved data items" as the first required data item of Q_i. We can formulate $p(j)$ as follows:

$$p(j) = \begin{cases} \frac{L - a(b) + a(1)}{L} & : \text{ if } j = 1, \\ \frac{a(j) - a(j-1)}{L} & : \text{ otherwise.} \end{cases} \qquad (4)$$

Denote the average retrieval time of Q_i as $T_{Retr.}(Q_i)$ and let $T^j_{Retr.}(Q_i)$ be the retrieval time of Q_i when $Cand(j)$ is the first retrieved data item. It is obvious that $T_{Retr.}(Q_i)$ can be obtained by the following equation.

$$T_{Retr.}(Q_i) = \sum_{j=1}^{b} p(j) \times T^j_{Retr.}(Q_i) \qquad (5)$$

Without loss of generality, we assume that the mobile device will retrieve $Cand(j)$ as the first retrieved data item at the m-th broadcast cycle. Also let the *retrieval point* represent the point that the mobile device starts to retrieve the first required data item of Q_i. Assume that $Cand(j)$ is the y-th copy of D_x (i.e., $Cand(j) = D_x(y)$). Therefore, the distance between the start time of the m-th broadcast cycle and the corresponding *retrieval point* is $offset(x, y)$. Since the mobile device will try its best to minimize the access time, for each data item D_k in Q_i, $k = q^i(1), q^i(2), \cdots, q^i(|Q_i|)$, the

mobile device will retrieve the first appeared copy of D_k after the *retrieval point*. To simplify the following derivation, we let $Point_{Retr.} = offset(x, y)$, and then, we have the following lemma.

Lemma 1 Let the function $NEAREST(Point_{Retr.}, k)$ represent the distance between the *retrieval point* and the point that the mobile device starts to retrieve D_k. The function $NEAREST(Point_{Retr.}, k)$ can be formulated as

$$NEAREST(Point_{Retr.}, k) = \begin{cases} \min\left\{offset(k,i) \middle| i = 1, 2, \cdots, n(D_k) \text{ and } offset(k,i) \geq Point_{Retr.}\right\} - Point_{Retr.} \\ \quad : \text{if there exists at least one copy of } D_k, \text{ say } D_k(r), \text{ where } offset(k,r) \geq Point_{Retr.} \\ L - Point_{Retr.} + \min\left\{offset(k,i) \middle| i = 1, 2, \cdots, n(D_y)\right\} \\ \quad : \text{otherwise.} \end{cases}$$

By Lemma 1, the distance between *retrieval point* and the point that the nearest copy of D_k has been completely retrieved is $NEAREST(Point_{Retr.}, k)+size(D_k)$. Since the mobile device will stop retrieving data items when all data items in Q_i have been completely retrieved, $T^j_{Retr.}(Q_i)$ can be formulated as

$$T^j_{Retr.}(Q_i) = \max_{k=q^i(1),q^i(2),\cdots,q^i(|Q_i|)} \left\{NEAREST(offset(Point_{Retr.}, k) + size(D_k)\right\} \times \frac{1}{B}. \quad (6)$$

Then, $T_{Retr.}(Q_i)$ can be obtained by Equations (5) and (6). By the definition of access time, $T_{Access}(Q_i)$ can be formulated as

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

$T_{Access}(Q_i)$ for $i = 1, 2, \cdots, |Q|$ can be obtained by the above equation, and finally, $T_{Access}(Q)$ can calculated by Equation (2).

4. DESIGN OF GENETIC ALGORITHM

As described before, fitness is the measurement of the quality of the chromosomes, and the GA is designed to search the chromosome with the highest fitness (i.e, maximize the fitness). Since the goal of broadcasting dependent data is to minimize the average access time of the given query profile, the fitness function is defined as

$$Fitness(P) = \frac{1}{T_{Access}(Q)}.$$

According to Equation (2), $T_{Access}(Q)$ is the weighted summation of all $T_{Access}(Q_i)$. Consequently, $T_{Access}(Q)$ can be obtained by the following procedure with the analytical results derived in Section 3.

Procedure CalAccessTime(Q, P)
Input: A query profile Q and a broadcast program P.
Output: $T_{Access}(Q)$ over the broadcast program P.

1: $T_{Access}(Q) \leftarrow 0$
2: **for** $i = 1$ to $|Q|$ **do**
3: $\quad T_{Access}(Q) \leftarrow T_{Access}(Q) + CalAccessTimeOfQuery(Q_i, P) \times Pr(Q_i)$
4: **end for**
5: return $T_{Access}(Q)$

Function CalAccessTimeOfQuery(Q_i, P)
Input: A query Q_i and a broadcast program P.
Output: $T_{Access}(Q_i)$ on the broadcast program P.

1: Calculate $T_{Startup}(Q_i)$ in accordance with the result of Equation (3)
2: Find the candidates of the first retrieved data items of Q_i (i.e., $Cand$) according to Definition 4.
3: $T_{Retr.}(Q_i) \leftarrow 0$
4: **for** $j = 1$ to $|Cand|$ **do**
5: $\quad$ Calculate $T^j_{Retr.}(Q_i)$ according to Equation (6)
6: $\quad$ Calculate $p(j)$ on the basis of Equation (4)
7: $\quad T_{Retr.}(Q_i) \leftarrow T_{Retr.}(Q_i) + p(j) \times T^j_{Retr.}(Q_i)$
8: **end for**
9: $T_{Access}(Q_i) \leftarrow T_{Startup}(Q_i) + T_{Retr.}(Q_i)$
10: return $T_{Access}(Q_i)$

Parameters	Values
The number of generations (n_{Gen})	20
The size of population (n_{Pop})	5
The probability of crossover (P_C)	0.9
The probability of mutation (P_M)	0.5
The size of a page	1K bytes
The bandwidth of each channel (B)	100K byte/sec.
The number of data items ($\|D\|$)	450
The replication factor $\left(\frac{size(D^*)}{size(D)}\right)$	2
The number of queries ($\|Q\|$)	300
The value of fanout	15
The Zipf distribution (θ)	2.5

Table 1: System parameters

5. PERFORMANCE EVALUATION

5.1 The Simulation Model

For performance studies, we implemented the GA-based algorithms with GAlib [5], and also a query profile generator. The simulator and query profile generator are both coded in C++. We define the replication factor as $\frac{size(D^*)}{size(D)}$ to represent the degree of replication. The probability of the query Q_i issued by users is assumed to be $Pr(Q_i) = \frac{(\frac{1}{i})^\theta}{\sum_{j=1}^{n}(\frac{1}{j})^\theta}$ where θ is the parameter of the Zipf distribution. Let the data size for each data item follows a normal distribution with mean 10 pages and variance 4 pages, and one page is set to be 1K bytes. Table 1 shows the system parameters in our experiments.

In addition to the proposed scheme, we also implement QEM [2] for comparison purposes. Note that QEM can generate broadcast programs for unordered queries without replication. To evaluate the effect of the proposed algorithm on the quality of solutions and the execution time, several experiments are conducted. For performance comparison, the performance gain of scheme A over scheme B is defined as

$$\frac{\text{Avg. access time of scheme A} - \text{Avg. access time of scheme B}}{\text{Avg. access time of the scheme B}}.$$

5.2 The Effect of the Number of Generations

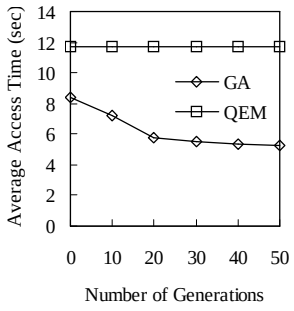

Figure 4: Avg. access time with the number of generations varied

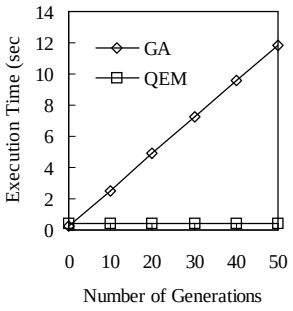

Figure 5: Exe. time with the number of generations varied

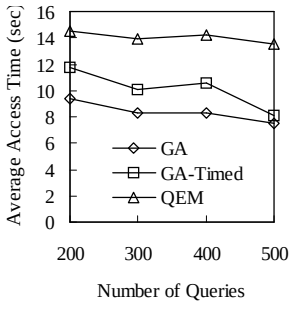

Figure 6: Avg. access time with number of queries varied

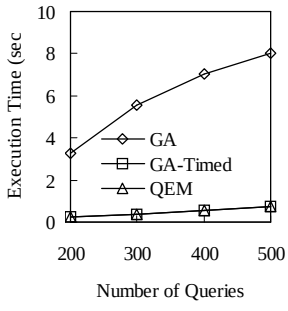

Figure 7: Exe. time with the number of queries varied

In the first experiment, we investigate the evolution process of the proposed GA by varying the value of n_{Gen}. Note the change of the value of n_{Gen} does not affect the results of scheme QEM. Figure 4 shows the effect of different values of n_{Gen} ranging from 0 to 50 on the average access times of the resulting broadcast programs. The corresponding execution times of the schemes are presented in Figure 5. Note that $n_{Gen} = 0$ represents the case of randomly generating n_{Pop} solutions. As observed in Figure 4, the average access time of the result broadcast program decreases as the value of n_{Gen} increases. However, the speed of convergence becomes slow when n_{Gen} is larger then 20. Therefore, we set n_{Gen} to be 20 in the following experiments. In addition, the performance gain of GA over QEM increases from 28% to 55% when n_{Gen} increase from 0 to 50.

As shown in Figure 5, the execution time of scheme GA increases linearly as the value of n_{Gen} increases. We also observe that scheme GA is about 30 times slower than scheme QEM when $n_{Gen} = 20$. However, scheme GA still outperforms scheme QEM when $n_{Gen} = 0$ (i.e., the execution of scheme GA is smaller than that of scheme QEM) which shows the importance of data replication in broadcast program generation.

5.3 The Effect of the Number of Queries

Figures 6 and 7 show the experimental results with the number of queries (i.e., $|Q|$) varied. As observed in Figure 6, the performance gain of GA over QEM ranges from 36% to 44% when the number of queries increases from 200 to 500. However, the performance gain of GA over GA-Timed decreases from 21% to 8%. It is because that a smaller number of queries indicates less constraints in broadcast program optimization, and therefore, GA outperforms GA-Timed since GA has more time in optimization than GA-Timed. On the other hand, when the number of queries is large, most effort in optimization will be in vain since the number of constraints is large. In addition, as observed in Figure 7, the execution times of scheme GA and QEM increase linearly as the number of queries increases.

6. CONCLUSION

We explored in this paper the problem of broadcasting dependent data for unordered queries and explicitly investigated the effect of data replication. By analyzing the model of dependent data broadcasting, we derived several theoretical properties for the average access time. In light of the theoretical results, we developed a genetic algorithms to generate broadcast programs for unordered queries with data replication. Our experimental results showed that with the analytical results derived, the theoretical results derived were able to guide the search of the genetic algorithm very effectively, thus leading to solution broadcast programs of very high quality. It is also shown by the experimental results that the data replication technique employed can lead to more efficient use of network bandwidth than prior schemes.

Acknowledgement

The authors are supported in part by the Ministry of Education Project No. 89E-FA06-2-4-7 and the National Science Council, Project No. NSC 91-2213-E-002-034 and NSC 91-2213-E-002-045, Taiwan, Republic of China.

7. REFERENCES

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast Disks: Data Management for Asymmetric Communication Environments. In *Proceedings of the ACM SIGMOD Conference*, pages 198–210, March 1995.

[2] Y. D. Chung and M. H. Kim. Effective Data Placement for Wireless Broadcast. *Distributed and Parallel Databases*, 9(2):133–150, March 2001.

[3] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, 1989.

[4] N. H. Vaidya and S. Hameed. Scheduling Data Broadcast in Asymmetric Communication Environments. *ACM Wireless Networks*, 5(3), May 1999.

[5] M. Wall. GAlib: A C++ Library of Genetic Algorithm Components. *http://lancet.mit.edu/ga*, August 1996.

[6] E. Yajima, T. Hara, M. Tsukamoto, and S. Nishio. Scheduling and Caching Strategies for Broadcasting Correlated Data. In *Proceedings of the ACM Symposium on Applied Computing*, pages 504–509, March 2002.