Multiple-Access Algorithms Via Group Testing for Heterogeneous Population of Users

DORON KURTZ AND MOSHE SIDI, SENIOR MEMBER, IEEE

Abstract—**The application of group testing techniques to the design of efficient algorithms for multiple-access systems with *heterogeneous* population of users, is studied. For a reservation system with binary (something/nothing) feedback, we present the procedure used to determine the optimal *nested* multiaccess algorithms. This procedure is not practical for moderate and large number of heterogeneous users because the computations and memory required for their determination are exponential in the number of users. Consequently, we introduce the notion of *ordered* algorithms and present the procedure for determining optimal ordered algorithm. The computational complexity and memory requirements of this procedure are polynomial in the number of users. To show the effectiveness of ordered algorithms, the performance of optimal ordered algorithms for various orders is studied and compared to the performance of simpler suboptimal ordered algorithms and with optimal nested algorithms.**

In addition, we study the performance of nested multiaccess algorithms for a direct transmission system with ternary feedback (that are based on group testing procedures) when the population of users is very large, and for simplicity, we consider a system with two classes of users.

I. Introduction

The application of group testing techniques to the design of efficient multiple-access algorithms has been the subject of several recent studies [1]–[3]. In all these studies it has been assumed that the population of users of the underlying multiple-access communication system is homogeneous, i.e., all users generate traffic with the same rate. In practice, however, we expect that different users will generate traffic with different rates. The goal of this paper is to propose and to analyze multiple-access algorithms, that are based on group testing techniques [4]–[8], for systems with heterogeneous population of users.

We consider both reservation and direct transmission systems for binary (Something/Nothing) and ternary (0, 1, e) channel feedback [1]. For each of these systems and feedback channels, we develop the procedures used to determine the optimal *nested* multiaccess algorithms when the population of users is heterogeneous (each user is active with probability p_i and idle with probability $q_i = 1 - p_i$). Optimal nested procedures are not practical for moderate and large number of heterogeneous users because the computations and the memory required for their determination grow very rapidly (exponentially) as the number of users grows. Consequently, we introduce the notion of *ordered* algorithms and present the procedures for determining optimal ordered algorithms. The computational complexity and the memory requirements of these procedures are polynomial in the number of users. To show the effectiveness of ordered algorithms, the performance of optimal ordered algorithms for various orders is studied and compared to the performance of simpler suboptimal ordered algorithms and with optimal nested algorithms (whenever it is possible to determine the latter). Due to space limitations, we present in this paper (Section III) the algorithms (and their analysis) only for a reservation system with binary feedback. For other combinations of systems and feedback channels, the interested reader is referred to [12].

Group testing procedures for heterogeneous population of users have been presented in [7]–[9]. In terms of multiaccess systems, these algorithms correspond to a reservation system with binary feedback. The optimal Dorfman-like algorithm (test a group and if it is active test each of its users sequentially) has been determined in [7]. Several examples of the optimal group testing algorithm when the probabilities p_i are restricted so that each q_i is an integer power of the largest q_i (say q), have been presented in [8]. The approach is to consider each user with probability $q_i = q^{d_i}$ (d_i integer) of being nondefective, as representing d_i users each with probability q of being nondefective, and then to apply known results for homogeneous population of users. However, even for this restricted set of probabilities, the procedure for determining the optimal algorithm has not been specified, except for the case that d_i can take only two values and the population of users is very large [9].

The work of Trisman [11] presents different kinds of multiaccess algorithms for a direct transmission system with ternary feedback when the p_i's can take only two values, i.e., there are only two types of users in the system. We consider such a system when the number of users is very large in Section IV.

II. The Multiple-Access System

Consider a finite population of N geographically distributed, bursty, independent users attempting to communicate with a central facility or with each other via a common channel. The users generate messages of fixed length called packets. Time is divided into slots of identical lengths that correspond to the transmission time of a packet. All users are synchronized and start the transmission of a packet at the beginning of a slot. Whenever two or more users transmit during the same slot, a collision occurs and none of the packets reaches its destination.

Two types of multiple-access systems are of interest: direct transmission systems and reservation systems. In a direct transmission system, whenever a user transmits a signal over the channel, that signal is the information packet itself. Consequently, collisions occur when two or more users transmit their packets during the same slot. Collided packets are destroyed and have to be retransmitted.

A reservation system has two stages. A first stage during which the users that have packets to transmit, are queried to make reservations, and a second stage during which users who have made reservations, transmit their packets. The time slot may be different for the two stages and collisions among users can occur only during the first stage.

We assume that at the end slot (for a reservation system at the end of each slot of the first stage), each user is informed about what happened during that slot via a common reliable feedback channel. We distinguish between two types of

Paper approved by the Editor for Random Access Systems of the IEEE Communications Society. Manuscript received March 19, 1987; revised March 3, 1988. This paper was presented at INFOCOM'87, San Francisco, CA, March 1987. This paper was based on the M.Sc. thesis of D. Kurtz.

The authors are with the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel.

IEEE Log Number 8824043.

feedback channels: binary feedback and ternary feedback. In a binary feedback channel, we assume that the users are informed whether the slot was empty (Nothing—no user transmitted) or not (Something—at least one user transmitted). This kind of feedback is called Something/Nothing (S/N) feedback. In a ternary feedback channel, we assume that the users are informed whether the slot was empty (LACK—no user transmitted), or contained a successful transmission (ACK—exactly one user transmitted), or contained a collision (NACK—at least two users transmitted). This kind of feedback is called 0, 1, e feedback.

The multiple-access algorithms that we propose operate cyclically; the collisions, if any, between packets that arrive during one cycle are resolved during the next cycle. For simplicity, we assume an i.i.d. model for generation of packets by the users. Specifically, we assume that the probability of user $i (1 \le i \le N)$ having a packet at the beginning of a cycle is p_i (the studies [1]–[3] assume that $p_i = p (1 \le i \le N)$, i.e., homogeneous population, an assumption that simplifies matters considerably). We say that a user is active (idle) in some cycle if it has (does not have) a packet at the beginning of that cycle.

The measure of performance for the proposed algorithms that is used for the reservation systems is the expected number of queries needed to identify all the active users during one cycle. For direct transmission systems, the performance measure that is used is the expected number of slots needed to transmit successfully the packets generated by active users during one cycle.

III. Reservation System with Binary (S/N) Feedback

In this section, we introduce and analyze multiaccess algorithms deploying S/N feedback for a reservation system. We first introduce the procedure for determining the optimal *nested* algorithm. Then we introduce the notion of *ordered* algorithms and show how to determine the optimal ordered algorithm. In addition, we introduce suboptimal ordered algorithms whose determination is much simpler than the optimal ones. In our reservation system model, when a subset of users is queried, all users in that subset wishing to make a reservation (all active users) respond by emitting a "1." The channel feedback indicates a "1" (S) if and only if at least one user responds by emitting a signal; else the feedback is "0" (N).

A general result for this system is stated in the following theorem (which extends the result of [10]).

Theorem 1: Assume a binary feedback reservation system with N users, $u_1, u_2, \cdots, u_N$ with probabilities $p_1, p_2, \cdots, p_N$, respectively, where $p_1 \le p_2 \le \cdots \le p_N$. If $2p_1 + p_2 - p_1p_2 > 1$ then the *optimal* algorithm is that each user is queried separately.[1]

The proof of this theorem appears in Appendix 1. For any other set of probabilities that do not satisfy the conditions of Theorem 1, we restrict ourselves to nested multiaccess algorithms [2], [4].

A. Nested Algorithms

Let Z denote the set of N users to be classified, let U (the unknown set) be a set of users where user i is active with probability p_i and is inactive with probability $q_i = 1 - p_i$. Let A (the active set) be a set of users known to contain at least one active user and let K (the known set) be the set of classified users. ($\varnothing$ will denote the empty set). The formal description of a general nested algorithm is presented in Algorithm 1. Note that the restriction of being nested appears in Step 3 in the algorithm where after a set A is found to be active, the next query is of a proper subset Y of A. Note also that in Step 3 of the algorithm we use the known fact [2], [4] that whenever a proper subset Y of an active set A is queried and the feedback is Something, the users in $A-Y$ can be returned to U since their distribution is the same as if they have never been queried.

```
Set $U \leftarrow Z$; $K \leftarrow \varnothing$; $A \leftarrow \varnothing$;
Step 0: If $U = \varnothing$, then END;
Step 1: Query a subset $X$ of $U$,
        If feedback is Nothing then $K \leftarrow K+X$; $U \leftarrow U-X$; Go to Step 0;
        If feedback is Something then $A \leftarrow X$; $U \leftarrow U-X$;
Step 2: If cardinality of $A = 1$ then $K \leftarrow K+A$; $A \leftarrow \varnothing$; Go to Step 0;
Step 3: Query a proper subset $Y$ of $A$;
        If feedback is Nothing then $K \leftarrow K+Y$; $A \leftarrow A-Y$; Go to Step 2;
        If feedback is Something then $U \leftarrow A-Y+U$; $A \leftarrow Y$; Go to Step 2;
```

Algorithm 1—A nested algorithm

B. Optimal Nested Algorithm

To determine the optimal nested algorithm, one has to decide upon the subsets X and Y in each corresponding step of the algorithm, so that the average number of queries needed to classify all users in Z will be minimized. These sets are determined by the following recursive relations that describe the optimal algorithm. Let $H(U)$ be the average number of additional queries required to complete the optimal algorithm when the active set is empty and the set U is the unknown set. Similarly, $G(U; A)$ is defined when A is not empty. Then (in the following $|U|$ is the cardinality of the set U):

$$H(\varnothing)=0;\ H(U)=1 \quad |U|=1 \tag{1a}$$

$$H(U)=1+\min_{\substack{X\subseteq U\\ X\neq\varnothing}} \{Q(X)H(U-X) + [1-Q(X)]G(U-X;\ X)\} \quad |U|\ge 2; \tag{1b}$$

$$G(U;\ A)=H(U) \quad |A|=1; \tag{1c}$$

$$G(U;\ A)=1+\min_{\substack{Y\subset A\\ Y\neq\varnothing}} \{PZ(A;\ Y)G(U;\ A-Y) + [1-PZ(A;\ Y)]G(U+A-Y;\ Y)\} \quad |A|\ge 2; \tag{1d}$$

where for any set D, $Q(D) = \Pi_{k\in D} q_k$ is the probability that none of the members of D is active and

$$PZ(A;\ Y)=\frac{Q(Y)-Q(A)}{1-Q(A)}. \tag{1e}$$

Equation (1a) gives the initial conditions and (1b)–(1d) describe Steps 1, 2, and 3 of the algorithm, respectively. The minimizing subsets X and Y obtained in (1b) and (1d) yield the optimum subsets to be queried in Steps 1 and 3 of Algorithm 1,

[1] A similar theorem has been recently proved (independently) in: Y. C. Yao and F. K. Hwang, "Individual testing of independent items in optimal group testing," *Prob. Eng. Inform. Sci.*, vol. 2, no. 1, pp. 23–29, 1988.

respectively. The average number of queries required to classify the set Z is given by $H(Z)$. For a small number of users the application of (1) is straightforward, hence, the determination of the optimal nested algorithm is fairly simple in this case.

C. Ordered Algorithms

For large, or even for moderate sets of users, the approach of finding an optimal nested algorithm is not practical. The reason is that the computational complexity (number of computations) and the memory size involved in determining an optimal nested algorithm [solving the recursive equations (1)] grow very rapidly (exponentially) as the number of users grows. For instance, in the first step of the algorithm, one needs to check $2^N - 1$ possible queries (the $2^N - 1$ ways of choosing the set X in Step 1). Furthermore, to determine the optimal nested algorithm for a set Z, one has to know the optimal nested algorithms for *all* proper subsets of Z.

Consequently, for moderate and for large number of users, we restrict ourselves to a subclass of nested algorithms, that we call *ordered* algorithms. Assume that at the beginning of the algorithm (the beginning of a cycle), the users to be classified (those in Z) are ordered in some order that remains fixed until the cycle ends. Without loss of generality, the first user in that order is called u_1, the second u_2, etc. Ordered algorithms are nested algorithms with the additional property, that the subset X (in Step 1) and the proper subset Y (in Step 3), always contain users with the least indexes in U and in A, respectively. Consequently, for ordered algorithms, one has only to specify *how many* users should X and Y contain in the respective steps (rather than *which* users).

The reason for considering ordered algorithms is twofold. First, it is possible to determine the optimal ordered algorithm (for a given order) via polynomial number of computations with memory whose size is polynomial in the number of users. Second, for small number of users, we observed that the performance of an optimal ordered algorithm is very close to that of optimal nested algorithms, and in many instances they coincide. In addition, note that ordered algorithms coincide with nested algorithms for homogeneous population.

D. Optimal Ordered Algorithm

Let $u_1, u_2, \cdots, u_N$ be the fixed order of the users. Let $H(u_i)$ $1 \le i \le N$ be the average number of additional queries needed to complete a cycle of the algorithm when the active set A is empty and the unknown set $U = (u_i \div u_N)$ [in this paper, the notation $B = (u_i \div u_j)$ indicates that the set B contains users $(u_i, u_{i+1}, \cdots, u_j)$]. Let $G(u_i \div u_j) 1 \le i \le N-1, i < j \le N$ be the average number of additional queries needed to complete a cycle of the algorithm when the active set $A = (u_i \div u_j)$ and the unknown set $U = (u_{j+1} \div u_N)$. Then the following recursions determine the optimal ordered algorithm:

$$H(u_{N+1}) = 0; \; H(u_N) = 1; \tag{2a}$$

$$H(u_i) = 1 + \min_{1 \le x \le N-i+1} \{Q(i, x)H(u_{i+x}) + (1 - Q(i, x)) \cdot G(u_i \div u_{i+x-1})\} \quad 1 \le i \le N-1; \tag{2b}$$

$$G(u_i \div u_i) = H(u_{i+1}) \quad 1 \le i \le N; \tag{2c}$$

$$G(u_i \div u_j) = 1 + \min_{1 \le y \le j-i} \{PZ(i, j, y)G(u_{i+y} \div u_j) + [1 - PZ(i, j, y)]G(u_i \div u_{i+y-1})\} \quad 1 \le i < j \le N; \tag{2d}$$

where

$$Q(i, x) = \prod_{k=i}^{i+x-1} q_k$$

and

$$PZ(i, j, y) = \left(\prod_{k=i}^{i+y-1} q_k - \prod_{k=i}^{j} q_k\right) \Big/ \left(1 - \prod_{k=i}^{j} q_k\right) \tag{2e}$$

When the unknown set contains only user u_N, it has to be queried, hence, $H(u_N) = 1$. The explanation of (2b) is as follows. When the unknown set $U = (u_i \div u_N)(1 \le i \le N - 1)$, and the active set is empty, then according to Step 1 of Algorithm 1, a subset X of U has to be queried. Let $x(1 \le x \le N - i + 1)$ be the cardinality of that set. Then $X = (u_i \div u_{i+x-1})$ since an ordered algorithm is used. The 1 in (2b) represents the query of the set X. If the feedback is Nothing [that happens with probability $Q(i, x)$] then all users in X have been classified and additional $H(u_{i+x})$ average number of queries are needed to complete the algorithm. Otherwise, the set X becomes active and additional $G(u_i \div u_{i+x-1})$ average number of queries are needed to complete the algorithm. The min operation in (2b) reflects the search for the optimal algorithm. Next, we note that (2c) describes Step 2 of Algorithm 1. Finally, (2d) describes Step 3 and its explanation is similar to that of (2b).

The order in which the quantities in (2) are calculated is $G(u_{N-1} \div u_N)$; $H(u_{N-1})$; $G(u_{N-2} \div u_{N-1})$; $G(u_{N-2} \div u_N)$; $H(u_{N-2})$; $\cdots$; $H(u_1)$. It is clear that $H(u_1)$ is the average number of queries of the optimal ordered algorithm. In the process of determining the optimal ordered algorithm the expressions in the curled braces in (2b) and in (2d) are calculated $O(N^2)$ times and $O(N^3)$ times, respectively. The required memory is $O(N)$ for H expressions and $O(N^2)$ for G expressions. Though being polynomial, for large number of users, $O(N^3)$ computations and $O(N^2)$ memory size may cause difficulties in determining the optimal ordered algorithm. Therefore, we introduce in the sequel a suboptimal ordered algorithm whose determination is rather simple.

E. Suboptimal Ordered Algorithm

The suboptimal ordered algorithm introduced here follows the steps of Algorithm 1, with the following rules for selecting the sets X in Step 1 and Y in Step 3.

Whenever the active set is empty and the unknown set contains users $U = (u_i \div u_N)$, then $X = (u_i \div u_{i+x-1})$ where x is given by

$$x = \arg \min_{1 \le x' \le N-i+1} \left| \prod_{k=i}^{i+x'-1} q_k - 0.5 \right| . \tag{3}$$

Whenever the active set $A = (u_i \div u_j)(j > i)$ and the unknown set $U = (u_{j+1} \div u_N)$, then $Y = (u_i \div u_{i+y-1})$ where y is given by

$$y = \arg \min_{1 \le y' \le j-i} \left| \prod_{k=i}^{i+y'-1} q_k - \prod_{k=i+y'}^{j} q_k \right| . \tag{4}$$

For homogeneous population, condition (4) is just a halving operation. For heterogeneous population it states that the active set is partitioned into two ordered subsets in such a way that the probabilities of having an active user are as close as possible in the two subsets. Condition (3) is an extension of the information algorithm of [4] to an environment of heterogeneous population.

We note that the computations of the set X in Step 1 and Y in Step 3 are simpler with the suboptimal algorithm and no memory is needed, hence their attractiveness. The performance of this algorithm is evaluated via the recursions (2) without the *min* operation in (2b) and (2d). Instead, x and y in (2b) and (2d) are obtained via (3) and (4), respectively. (We reemphasize that these recursions are not needed for the determination of the suboptimal algorithm.)

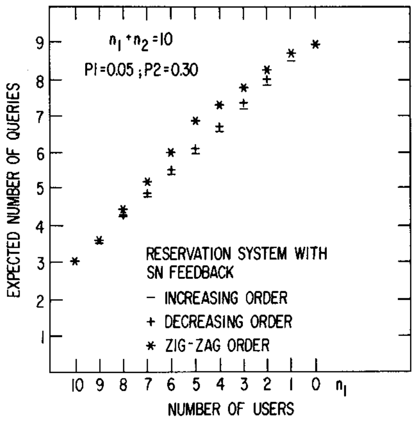


Fig. 1. Reservation system with binary feedback.

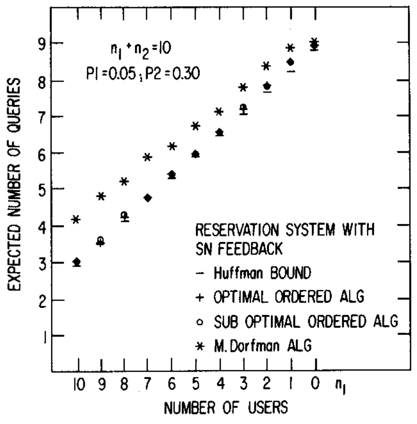


Fig. 2. Reservation system with binary feedback.

F. Numerical Results

In general, the performance of the optimal and suboptimal ordered algorithms will depend on the specific order among the users that is employed. Determination of the optimal order needs an exponential (in the number of users) number of computations and therefore is not practical. Henceforth, we will confine ourselves to three different orders. We define an *increasing order* as an order with $p_1 \leq p_2 \leq \cdots \leq p_N$ and a *decreasing order* as an order with $p_1 \geq p_2 \geq \cdots \geq p_N$. For a two-class system (n_1 users each of which is active with probability $P1$ and n_2 users each of which is active with probability $P2$; $n_1 + n_2 = N$), we also define a *zig-zag* order as an order for which u_1 is taken from the first class, u_2 from the second class, u_3 from the first class, etc., until there are no more users of one of the classes, in which case the rest of the users belong to the other class.

Consider a two-class system with $P1 = 0.05$ and $P2 = 0.3$. Fig. 1 shows the effect of various orders on the optimal ordered algorithm when $n_1 + n_2 = 10$. From this figure and from many other examples, we found that in a reservation system with binary feedback, the increasing order is the better order for a wide range of parameters. Yet, we note that the performance of the optimal ordered algorithm is not that sensitive to the order employed. Fig. 2 shows the performance of the optimal and suboptimal ordered algorithms (with increasing order) along with the lower bound on the number of queries as derived via the Huffman code (for details on this bound, see [2], [8, p. 277], and [14]) and along with the modified Dorfman algorithm [7]. We notice that the performance of the optimal ordered algorithm does not fall very much behind the lower bound (recall that the lower bound is not always achievable [2]), and the performance of the suboptimal ordered algorithm is very close to that of the optimal ordered algorithm. Both are better than the modified Dorfman algorithm. The same kind of behavior has been observed in many other examples. Fig. 3 shows the performance of the optimal ordered algorithm as the number of users grow ($n_1 = n_2$) for three cases: $P1 = P2 = 0.05$, $P1 = P2 = 0.01$, $P1 = 0.01$ and $P2 = 0.05$. For the latter case, the performance of the suboptimal ordered algorithm, the entropy lower bound (see [8], [14]) and the performance of an algorithm that deals separately with each class of users, are also presented.

To get some feeling of the performance of these algorithms when more classes of users are present, consider a system with 4 users each is active with probability $P1 = 0.1$, 3 users each is active with probability $P2 = 0.3$, and 3 users each is active with probability $P3 = 0.5$. For this system, the average number of queries required when the optimal ordered algorithm (increasing order) is employed is 7.6078, while the suboptimal algorithm requires 7.6105 queries (the lower bound is 7.564). For the same system with $P1 = 0.05$, $P2 = 0.08$, and $P3 = 0.1$ the corresponding numbers are 3.9, 4.2 (3.79).

IV. Very Large Population

The purpose of this section is to study the performance of nested multiaccess algorithms (that are based on group testing procedures) when the population of users is very large. We restrict ourselves to a direct transmission system with ternary feedback and for simplicity, we consider a system with two classes of users. Specifically, we assume that there are M_1 users each with probability of being active p_1 and M_2 users each with probability of being active p_2 where $M_1 \simeq M_2$ and both M_1 and M_2 are very large.

```
Set $U \leftarrow Z$; $K \leftarrow \emptyset$; $A \leftarrow \emptyset$; $C \leftarrow \emptyset$;
Step 0: If $U = \emptyset$, then END;
Step 1: Let the users of a subset $X$ of $U$ transmit,
        If feedback is LACK or ACK then $K \leftarrow K+X$; $U \leftarrow U-X$; Goto Step 0;
        If feedback is NACK then $C \leftarrow X$; $U \leftarrow U-X$;
Step 2: Let the users of a proper subset $Y$ of $C$ transmit,
        If feedback is LACK then $K \leftarrow K+Y$; $C \leftarrow C-Y$; Goto Step 2;
        If feedback is ACK then $K \leftarrow K+Y$; $A \leftarrow C-Y$; $C \leftarrow \emptyset$; Goto Step 3;
        If feedback is NACK then $U \leftarrow U+C-Y$; $C \leftarrow Y$; Goto Step 2;
Step 3: Let the users of a subset $W$ of $A$ transmit,
        If feedback is LACK then $K \leftarrow K+W$; $A \leftarrow A-W$; Goto Step 3;
        If feedback is ACK then $K \leftarrow K+W$; $U \leftarrow U+A-W$; $A \leftarrow \emptyset$; Goto Step 0;
        If feedback is NACK then $C \leftarrow W$; $U \leftarrow U+A-W$; Goto Step 2;
```

Algorithm 2—A nested algorithm.

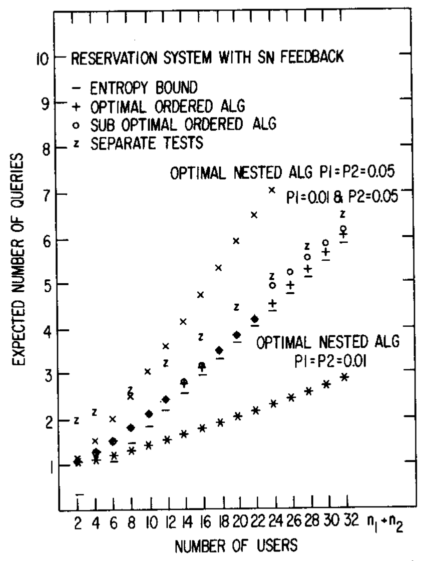


Fig. 3. Reservation system with binary feedback.

The description of a nested multiple access algorithm for a direct transmission system with ternary feedback appears in Algorithm 2. Each visit to Step 1 is called an H-visit. The measure of performance that is used for a very large population of users is the *throughput*—the average number of successful transmissions per slot.

A. Separate Test Algorithm

The first algorithm that we consider is an algorithm that handles each class of users separately and independently of the other class. We call such an algorithm a *separate test algorithm*. In this algorithm, we first classify all users of the first class and then the users of the other class (other ways to classify the users of the two classes separately are discussed at the end of this section). For this algorithm, let $H_i(x_i)(i = 1, 2)$ be the average number of slots between two successive H-visits, when in Step 1 of the algorithm we let x_i users (from class i) to transmit, and let $N_i(x_i)$ be the average number of users that are classified (either known to be idle or transmit their packets successfully) during $H_i(x_i)$. Then the throughput of this algorithm is given by

$$T_1 = \frac{M_1p_1 + M_2p_2}{\dfrac{M_1H_1(x_1)}{N_1(x_1)} + \dfrac{M_2H_2(x_2)}{N_2(x_2)}}. \tag{5}$$

The reason for (5) is clear. The throughput is the average number of packets in the system divided by the average number of slots required to transmit these packets. The average number of packets in the system is $M_1p_1 + M_2p_2$. For users of class $i(i = 1, 2)$, we classify $N_i(x_i)$ users (on the average) between two successive H-visits, hence, (M_i is very large and therefore edge effects can be neglected) we need $M_i/N_i(x_i)$ H-visits (on the average) to classify all users of class i. The average number of slots between two successive H-visits is $H_i(x_i)$, and hence (5). Since $M_1 \simeq M_2$, we have

$$T_1 = \frac{p_1 + p_2}{\dfrac{H_1(x_1)}{N_1(x_1)} + \dfrac{H_2(x_2)}{N_2(x_2)}}. \tag{6}$$

Let C_1 be the maximal throughput that can be achieved with a separate test algorithm. Then

$$C_1 = \max_{x_1, x_2 > 0} T_1 = \frac{p_1 + p_2}{\min\limits_{x_1>0} \dfrac{H_1(x_1)}{N_1(x_1)} + \min\limits_{x_2>0} \dfrac{H_2(x_2)}{N_2(x_2)}}. \tag{7}$$

In other words, for given p_1 and p_2, we have to determine (separately and independently) x_1 and x_2 (and the corresponding algorithms). To that end, we can use the method presented in [13] where the optimal nested algorithm (that maximizes the throughput) for infinite homogeneous population, has been determined.

B. Joint Test Algorithm

The second algorithm that we consider is an algorithm that jointly operates on the two classes of users. We call this algorithm a *joint test algorithm*. In this algorithm the set X of users that is chosen in Step 1 of algorithm 2 contains x_1 users of the first class and x_2 users of the second class ($x_1 + x_2 > 0$), except when there are no longer users of one of the classes. In the latter case, the remaining users (from the other class) are handled as in the separate test algorithm. The reason that there are such remaining users is that the joint test algorithm does not necessarily classify users of both classes in the same rate. Note that if either $x_1 = 0$ or $x_2 = 0$ then this algorithm degenerates to the separate test algorithm.

For this algorithm, let $H(x_1, x_2)$ be the average number of slots between two successive H-visits, when in Step 1 of the algorithm we let x_1 users (from class 1) and x_2 users (from class 2) transmit, and let $N_i(x_1, x_2)(i = 1, 2)$ be the average number of users of class i that are classified (either known to be idle or transmit their packets successfully) during the $H(x_1, x_2)$ slots. Without loss of generality, let us assume that $N_1(x_1, x_2) \geq N_2(x_1, x_2)$. Then the throughput of this algorithm is given by

$$T_2 = \frac{M_1p_1 + M_2p_2}{\dfrac{M_1H(x_1, x_2)}{N_1(x_1, x_2)} + \left[M_2 - \dfrac{M_1N_2(x_1, x_2)}{N_1(x_1, x_2)}\right] \dfrac{H_2(x_2^*)}{N_2(x_2^*)}}. \tag{8}$$

The nominator in (8) is the same as in (5), while the average number of slots required to transmit all packets is different from that in (5). We start with letting x_1 and x_2 users transmit. Assuming that the rate of classifying users of the first class is higher than the rate of classifying users of the second class, we have that the number of H-visits is $M_1/N_1(x_1, x_2)$ (on the average), and the number of slots required to classify all class 1 users is $M_1H(x_1, x_2)/N_1(x_1, x_2)$ (on the average). We are left with $M_2^* = M_2 - M_1N_2(x_1, x_2)/N_1(x_1, x_2)$ users of the second class yet to be classified. These users are classified separately by the separate test algorithm where we denoted by x_2^* the number of users that transmit in Step 1 of the algorithm in this situation

Consequently, for $M_1 \simeq M_2$ and $N_1(x_1, x_2) \geq N_2(x_1, x_2)$, we have

$$T_2 = \frac{p_1 + p_2}{\dfrac{H(x_1, x_2)}{N_1(x_1, x_2)} + \left[1 - \dfrac{N_2(x_1, x_2)}{N_1(x_1, x_2)}\right] \dfrac{H_2(x_2^*)}{N_2(x_2^*)}}. \tag{9}$$

Let C_2 be the maximal throughput that can be achieved with a joint test algorithm. Let $f(p_2) = \min_{x_2^*>0} H_2(x_2^*)/N_2(x_2^*)$ (as was said before, $f(p_2)$ is calculated using the method in [13]). Then

$$C_2 = \frac{p_1 + p_2}{\min\limits_{x_1>0, x_2} \left\{\dfrac{H(x_1, x_2)}{N_1(x_1, x_2)} + \left[1 - \dfrac{N_2(x_1, x_2)}{N_1(x_1, x_2)}\right] f(p_2)\right\}}. \tag{10}$$

For any given joint test algorithm, the quantities $H(x_1, x_2)$ and $N_i(x_1, x_2)(i = 1, 2)$, can be calculated as follows. Let A be a set of users known to contain at least one active user. Let

TABLE I

Probabilities		Separate			tests		Joint			test	
p_1	p_2	k_1	k_2	$N_1(k_1)$	$N_2(k_2)$	C_1	x_1	x_2	$N_1(x_1,x_2)$	$N_2(x_1,x_2)$	C_2
0.05	0.1	26	13	23.797	12.000	0.5269	7	9	6.554	8.261	0.5290
0.05	0.3	26	4	23.797	3.820	0.5871	5	3	4.929	2.869	0.5896
0.05	0.6	26	2	23.797	2.000	0.6791	8	1	7.934	1.000	0.6835
0.05	0.725	26	1	23.797	1.000	0.7064	1	1	1.000	1.000	0.7226
0.1	0.2	13	6	12.000	5.688	0.5533	4	4	3.748	3.748	0.5583
0.1	0.3	13	4	12.000	3.820	0.5826	3	3	2.936	2.872	0.5849
0.1	0.4	13	3	12.000	3.000	0.6023	3	2	2.920	1.960	0.6124
0.1	0.5	13	2	12.000	2.000	0.6400	2	2	2.000	2.000	0.6451
0.1	0.8	13	1	12.000	1.000	0.7579	1	1	1.000	1.000	0.7759
0.2	0.25	6	4	5.688	3.875	0.5732	2	3	1.875	2.938	0.5763
0.2	0.4	6	3	5.688	3.000	0.6016	2	2	1.920	1.920	0.6117
0.2	0.5	6	2	5.688	2.000	0.6336	3	1	3.000	1.000	0.6371
0.2	0.6	6	2	5.688	2.000	0.6586	2	1	2.000	1.000	0.6700
0.3	0.4	4	3	3.820	3.000	0.6131	1	2	1.000	2.000	0.6154
0.3	0.8	4	1	3.820	1.000	0.7338	1	1	1.000	1.000	0.7432

C be a set of users known to contain at least two active users. Let $F(m_1, m_2)$ be the average number of slots until the next H-visit when A contains m_1 users from the first class and m_2 users from the second class, and $NF_i(m_1, m_2)$ be the average number of class i users that are classified during the $F(m_1, m_2)$ slots. Similarly, $E(m_1, m_2)$ and $NC_i(m_1, m_2)$ are defined when the set C contains m_1 and m_2 users. Let x_i, y_i and w_i be the number of users of class i that are allowed to transmit in Steps 1, 2, and 3 of the algorithm (see Algorithm 2), respectively. Then the quantities $H(x_1, x_2)$ and $N_i(x_1, x_2)(i = 1, 2)$ are calculated via the recursive equations that appear in Appendix 2. The joint test algorithm that maximizes the throughput can now be determined by exhaustive search over the parameters x_1, x_2, y_1, y_2, w_1, w_2. To limit the number of searches, we restricted ourselves to algorithms in which $w_i = m_i (i = 1, 2)$ whenever the active set contains m_1 users of the first class and m_2 users of the second class.

The results are summarized in Table I where the throughputs obtained with the separate test algorithm and the joint test algorithm are indicated along with the sizes of the sets x_1 and x_2 that should be chosen in H-situations. Evidently, the joint test algorithm is only slightly better than the separate test algorithm. The difference between the throughputs of the two tests is usually within 0–1 percent and it does not exceed 2.5 percent. Practically, this implies that the separate test (that is simpler) should be preferred.

Finally, note that both the separate test and the joint test (as they were described earlier) are not fair, in the sense of classifying all the users of one class before many of the users of the other class. This can be simply overcome, by alternating between the two classes in the separate test (according to the rate of classifying the users) or by alternating between the joint test and the separate test in the joint test algorithm.

APPENDIX I

Proof of Theorem 1: The proof of this theorem is similar to that in [10] (for the homogeneous population).

Multiaccess algorithm deploying S/N feedback can be represented by a binary tree. Each node of the tree corresponds to a query of a particular subset of the users (the label of the node is the subset), and has two arcs: the left is labeled N (Nothing) and the right is labeled S (Something). We say that two subsets occur on the same branch of a tree if they occur at two nodes, one of which can be reached from the other by only descending through the tree. The following Lemma is obvious.

Lemma 1: In the optimal algorithm a subset of users is not queried if its feedback indication can be inferred from previous queries.

Consequently the optimal algorithm has the following properties:

a) Subset G will not occur twice on the same branch, i.e., we need not query subsets already queried.

b) Let G be the queried subset at node B of the tree. No user of G occurs in any of the subsets on the "Nothing" branch starting at B, i.e., noting is gained by adding users known to be idle to any subset.

c) Let G be the queried subset at node B of the tree. Then any subset $G' \supset G$ occurs on either branch beneath G (if G is active then G' is known to be active and otherwise see b).

If an algorithm does not satisfy these three properties it can be modified into one which does, by removing classified users and omitting unnecessary queries.

We prove the theorem by taking an arbitrary algorithm which satisfies a)–c) and contains a subset of more than one user and modifying it so that the expected number of queries in the new algorithm is less than in the original one when the conditions of the theorem hold. This will show that an algorithm satisfying a)–c) which contains queries of more than 1 user cannot be optimal.

Let B be the node on our binary test tree where subset G is queried. G has m users where $m \geq 2$ and all the subsets on the branches beneath B contain one user. Denote the branch to be followed in case G is idle by I and otherwise by II. Let u_j denote any user in G. In the new algorithm, instead of querying G when we reach B we query $G - \{u_j\}$. If the feedback indicates an S, we continue the algorithm in the same manner as when G was found active in the old algorithm and omitting unnecessary queries. We denote this branch by II'. If the feedback indicates N we query $\{u_j\}$. If the feedback indicates N, we have exactly the same information as when G was found idle in the old algorithm, and we continue our algorithm in the same manner. If the feedback indicates $S(u_j$ active), we continue as in the case G is active in the old algorithm skipping queries the result of which can be inferred. We denote this branch by II''. The remainder of the algorithm, i.e., everything which is not below B is left unchanged.

The number of queries in the new algorithm can exceed the number of queries in the old algorithm by at most one. Furthermore, we have the following.

Lemma 2: The only samples for which one more query is needed under the new algorithm are those for which node B is reached and G is found idle under the old algorithm.

Proof: If $G - \{u_j\}$ is active we follow the same procedure except for possibly skipping a query which previously had to be performed. Otherwise, if $G - \{u_j\}$ is idle and user u_j is active, we enter II' having performed one more query than when we entered II under the old algorithm. However, under the old algorithm the idleness of the $(m - 1)$ users of $G - \{u_j\}$ must be ascertained by $m - 1$ queries along the branch II which can now be skipped. Since $m \geq 2$, this proves Lemma 2.

Let A denote a sample for which one more query is required under the new algorithm (all the users of G are idle). From properties a)–c) it follows that if for this case we reach node B then when some of the users of G are active we also reach B. Furthermore, the classification of the users of G, in the old and in the new algorithms, must be found from queries of G and $G - \{u_j\}$, respectively, and subsequent queries (each user is queried separately). Now we distinguish between the cases $m = 2$ and $m > 2$.

When $m = 2$ the subset G contains two users u_i and u_k (assume $p_i \leq p_k$). Under the old algorithm, if G is active, u_i is queried before u_k (this can be immediately checked). The following two cases require one less query under the new algorithm: 1) both users are active, 2) user u_i is active and user u_k is idle. Consequently, the expected number of tests saved when $m = 2$ is thus at least

$$\text{Prob}\,\{A\}\left[\frac{p_i p_k}{q_i q_k}+\frac{p_i}{q_i}-1\right]. \tag{A1}$$

Since $p_1 \leq p_i$ and $p_2 \leq p_k$ and $2p_1 + p_2 - p_1 p_2 > 1$, it follows that $2p_i + p_k - p_ip_k > 1$ and hence the expression in the braces in (A1) is positive.

When $m > 2$ there is a net saving of at least $m - 2$ queries when $G - \{u_j\}$ is idle and user u_j is active. Indeed, under the old algorithm, we need at least $(m - 1)$ individual queries to ascertain that all the users of $G - \{u_j\}$ are idle. These queries can be omitted under the new algorithm. Next, let u_l be the last user of $G - \{u_j\}$ to be queried when we apply the old algorithm. For the case for which only u_j and u_l are active we save one query under the new algorithm. (The fact that u_l is active can be ascertained by the fact that $(m - 1)$ users of $G - \{u_j\}$ are idle). Consequently, for m $\geq$ 3 the expected number of queries saved is at least

$$\text{Prob }\{A\}\left[(m-2)\frac{p_j}{q_j}+\frac{p_jp_l}{q_jq_l}-1\right]. \qquad \text{(A2)}$$

The expression in the braces is positive if

$$(m-1)p_j-(m-2)p_jp_l+p_l>1. \qquad \text{(A3)}$$

Straightforward algebra shows that (A3) holds true in both cases $p_j \leq p_l$ (we use the fact that $2p_j + p_l - p_jp_l > 1$), and $p_l \leq p_j$ (we use the fact that $2p_l + p_j - p_jp_l > 1$).

References

[1] T. Berger, N. Mehravari, D. Towsley, and J. K. Wolf, "Random multiple access communication and group testing," *IEEE Trans. Commun.*, vol. COM-32, pp. 769-778, July 1984.

[2] J. K. Wolf, "Born again group testing: Multiaccess communications," *IEEE Trans. Inform. Theory,* vol. IT-31, pp. 185-191, Mar. 1985.

[3] N. Seshadri and P. R. Srikartakumar, "On group testing protocols for binary-feedback multiaccess channel," *IEEE Trans. Commun.*, vol. COM-33, pp. 574-575, June 1985.

[4] M. Sobel and P. A. Groll, "Group testing to eliminate efficiently all defective in binomial sample," *Bell Syst. Tech. J.*, vol. 38, pp. 1178-1252, Sept. 1959.

[5] M. R. Garey and F. K. Hwang, "Isolating a single defective using group testing," *J. Amer. Statist. Ass.*, vol. 69, pp. 151-153, Mar. 1974.

[6] F. K. Hwang, "On finding single defective in binomial group testing," *J. Amer. Statist. Ass.*, vol. 69, pp. 146-150, Mar. 1974.

[7] ——, "A generalized binomial group testing problem," *J. Amer. Statist. Ass.*, vol. 70, pp. 923-926, Dec. 1975.

[8] E. Nebenzahl and M. Sobel, "Finite and infinite models for generalized group testing with unequal probabilities of success for each item," in *Discriminant Analysis and Application,* T. Cacoullos Ed. New York: Academic, 1973, pp. 239-289.

[9] E. Nebenzahl, "Binomial group testing with two different success parameters," *Stud. Sci. Math. Hunga.*, vol. 10, pp. 61-75, 1975.

Appendix II

$$H(0,0)=0;\ H(x_1,x_2)=1\quad x_1+x_2=1;\ H(x_1,x_2)=1+PC_1E(x_1,x_2)\quad x_1+x_2\geq 2$$

$$E(m_1,m_2)=0\quad m_1+m_2\leq 1;\ E(m_1,m_2)=2\quad m_1+m_2=2$$

$$E(m_1,m_2)=1+PZ_2E(m_1-y_1,m_2-y_2)+PS_2F(m_1-y_1,m_2-y_2)+PC_2E(y_1,y_2)\quad m_1+m_2\geq 3$$

$$F(m_1,m_2)=1\quad m_1+m_2=1;\ F(m_1,m_2)=1+PZ_3F(m_1-w_1,m_2-w_2)+PC_3E(w_1,w_2)\quad m_1+m_2\geq 2$$

$$N_i(x_1,x_2)=1\quad x_1+x_2=1,\ i=1,2;\ N_i(x_1,x_2)=x_i(PZ_1+PS_1)+PC_1NC_i(x_1,x_2)\quad x_1+x_2\geq 2,\ i=1,2$$

$$NC_i(m_1,m_2)=m_i\quad m_1+m_2=2,\ i=1,2;\ NC_i(m_1,m_2)=PZ_2[y_i+NC_i(m_1-y_1,m_2-y_2)]$$
$$+PC_2NC_i(y_1,y_2)+PS_2[y_i+NF_i(m_1-y_1,m_2-y_2)]\quad m_1+m_2\geq 3,\ i=1,2$$

$$NF_i(m_1,m_2)=m_i\quad m_1+m_2=1,\ i=1,2$$

$$NF_i(m_1,m_2)=PZ_3[w_i+NF_i(m_1-w_1,m_2-w_2)]+PS_3w_i+PC_3NC_i(w_1,w_2)\quad m_1+m_2\geq 2,\ i=1,2$$

where PZ_l, PS_l, and PC_l, l = 1, 2, 3 are given in the following (here $q_i = 1-p_i$ and $Q_i(x_i) = q_i^{x_i}$):

$$PZ_1=Q_1(x_1)Q_2(x_2);\ PS_1=x_1p_1Q_1(x_1-1)Q_2(x_2)+x_2p_2Q_2(x_2-1)Q_1(x_1)$$

$$PZ_2=\frac{Q_1(y_1)Q_2(y_2)\{1-Q_1(m_1-y_1)Q_2(m_2-y_2)-(m_1-y_1)p_1Q_1(m_1-y_1-1)Q_2(m_2)\}}{1-Q_1(m_1)Q_2(m_2)-m_1p_1Q_1(m_1-1)Q_2(m_2)-m_2p_2Q_2(m_2-1)Q_1(m_1)}$$
$$-\frac{Q_1(y_1)Q_2(y_2)(m_2-y_2)p_2Q_2(m_2-y_2-1)Q_1(m_1)}{1-Q_1(m_1)Q_2(m_2)-m_1p_1Q_1(m_1-1)Q_2(m_2)-m_2p_2Q_2(m_2-1)Q_1(m_1)}$$

$$PS_2=\frac{[y_1p_1Q_1(y_1-1)Q_2(y_2)+y_2p_2Q_2(y_2-1)Q_1(y_1)][1-Q_1(m_1-y_1)Q_2(m_2-y_2)]}{1-Q_1(m_1)Q_2(m_2)-m_1p_1Q_1(m_1-1)Q_2(m_2)-m_2p_2Q_2(m_2-1)Q_1(m_1)}$$

$$PZ_3=\frac{Q_1(w_1)Q_2(w_2)\{1-Q_1(m_1-w_1)Q_2(m_2-w_2)\}}{1-Q_1(m_1)Q_2(m_2)}$$

$$PS_3=\frac{w_1p_1Q_1(w_1-1)Q_2(w_2)+w_2p_2Q_2(w_2-1)Q_1(w_1)}{1-Q_1(m_1)Q_2(m_2)}$$

$$PC_l=1-PZ_l-PS_l\quad l=0,1,2.$$

[10] P. Ungar, "The cut off point for group testing," *Commun. Pure Appl. Math.*, vol. 13, pp. 49-54, Feb. 1960.

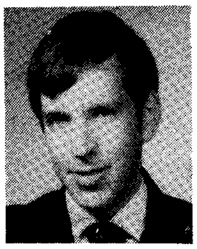
[11] R. C. Trisman, "A random access channel with two types of users," Master thesis, Elec. Eng. Dept., Univ. Massachusetts, Sept. 1983.

[12] M. Sidi and D. Kurtz, "Multiple access algorithms via group testing for heterogeneous population of users," presented at INFOCOM'87, San Francisco, CA, Apr. 1987.

[13] S. Panwar, D. Towsley, and J. K. Wolf, "On the throughput of degenerate intersection and first-come first-served collision resolution algorithms," *IEEE Trans. Inform. Theory,* vol. IT-31, pp. 274-279, Mar. 1985.

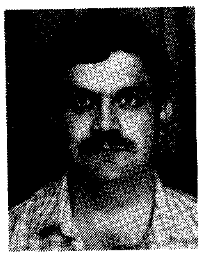
[14] M. Sobel, "Group testing to classify efficiently all units in a bionomial sample," in *Information and Decision Processes,* R. Machol Ed. New York: McGraw-Hill, 1960, pp. 127-161.

★



Doron Kurtz received the B.Sc. and M.Sc. degrees in electrical engineering from the Technion—Israel Institute of Technology, Haifa, Israel, in 1978 and 1986, respectively.

Since 1978 he has served in the I.D.F. Signal Corps Research laboratories and is now the Head of a research and development group. He has been involved in research and implementation of state-of-the-art systems.



Moshe Sidi (S'77–M'82–SM'87) received the B.Sc., M.Sc., and D.Sc. degrees from the Technion—Israel Institute of Technology, Haifa, Israel, in 1975, 1979, and 1982, respectively, all in electrical engineering.

From 1975 to 1981 he was a Teaching Assistant and a Teaching Instructor at the Technion in communication and data networks courses. In 1982 he joined the Faculty of the Electrical Engineering Department at the Technion. During the Academic year 1983–1984 he was a Post-Doctoral Associate at the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology, Cambridge, MA. During the academic year 1986–1987 he was a Visiting Scientist at IBM Thomas J. Watson Research Center, Yorktown Heights, NY. His current research interests are in queueing systems and in the area of computer communication networks.