# Finding Extrema With Unary Predicates

by

Feng Gao, Leonidas J. Guibas, David G. Kirkpatrick,
William T. Laaser, James Saxe

TR 90-39

Technical Report 90-39
January, 1991

# Finding Extrema With Unary Predicates

Feng Gao * Leonidas J. Guibas † David G. Kirkpatrick ‡
William T. Laaser § James Saxe ¶

**Abstract**

We consider the problem of determining the maximum and minimum elements of a set $X = \{x_1, \ldots, x_n\}$, drawn from some finite universe $\mathcal{U}$ of real numbers, using only unary predicates of the inputs. It is shown that $\Theta(n + log|\mathcal{U}|)$ unary predicate evaluations are necessary and sufficient, in the worst case. Results are applied to i) the problem of determining approximate extrema of a set of real numbers, in the same model, and ii) the multiparty broadcast communication complexity of determining the extrema of an arbitrary set of numbers held by distinct processors.

## 1 Introduction

A familiar guessing game has one player $A$ choose a real number $x$, from a given finite universe $\mathcal{U}$, while a second player $B$ attempts to determine $x$ by a short sequence of yes-no questions. It is widely appreciated (even by those who are not computer scientists) that a guessing strategy exists for $B$ which is guaranteed to identify the chosen number with at most $\log_2 | \mathcal{U} |$ questions[1]. The optimality of this strategy is easily established by an elementary adversary argument (*cf.* [1]); in effect $A$ tries to postpone her specific choice as long as possible.

We are interested in a generalization of this game in which some number $n \geq 1$ players $A_1, \cdots, A_n$ choose real numbers $x_1, \cdots, x_n$ respectively while a last player $B$ attempts to determine some function $f(X)$ of $X = (x_1, x_2, \cdots, x_n)$ by means of yes-no questions directed to $A_1, \cdots, A_n$ individually. It follows from the conventional game that $B$ can determine all of the values $x_1, \cdots, x_n$ (and thereafter, $f(X)$) in at most $n \log | \mathcal{U} |$ questions. However, the evaluation of $f$ may not require exact knowledge of all of its arguments and hence more efficient guessing strategies may exist for specific functions of interest. In this paper we show that this is the case for the functions *max* and *min*; their complexities in this unary predicate evaluation model are shown to be $\Theta(n + \log | \mathcal{U} |)$.

Before setting out our specific results in more detail it is worth describing some of the motivation for our study. The multiplayer guessing game models in a natural way

*Department of Computer Science, University of British Columbia, Vancouver, BC V6T 1W5, Canada.
†Laboratory for Computer Science, MIT, Cambridge, MA 02139, U.S.A., and DEC Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301, U.S.A.
‡Department of Computer Science, University of British Columbia, Vancouver, BC V6T 1W5, Canada.
§Metaphor Computer Systems, 1965 Charleston Road, Mountainview, CA 94043, U.S.A.
¶DEC Systems Research Center, 130 Lytton Avenue, Palo Alto, CA 94301, U.S.A.
[1]Hereafter all logarithms have base 2.

the evaluation of functions in two realistic computational settings. The first involves a situation in which the function arguments are provided by independent black boxes. Since the arguments are not available explicitly they can not be directly compared. Such is the case, for example, if $g_1 \cdots g_n$ is a sequence of strictly convex functions on $\mathcal{U}$ and $x_i$ is defined by the equation $g_i(x_i) = \max_{x \in \mathcal{U}} g_i(x)$. In this situation, it is reasonable to restrict algorithms to the use of (possibly constrained) unary predicates of the $x_i$'s.

The second setting captured by the multiplayer guessing game involves cooperative distributed computation. In this case, the arguments $x_1, \cdots, x_n$ are known individually and explicitly to $n$ distinct processors $P_1, \cdots, P_n$ respectively, and the objective is to collectively determine $f(X)$ by using the least amount of broadcast (one-to-all) communication. In this case individual processors can communicate information based on their evaluation of arbitrary unary functions of their own input, together with earlier communications. Each bit of communication of processor $P_i$ can be expressed as a unary predicate of $x_i$. Our model for multiparty communication complexity is a natural extension of the two-party model of Yao [9]. It closely resembles the multiparty models of [2] and [3] in which parties communicate through a shared "blackboard". Our model differs from these in that we assume no sharing of input values.

A straightforward information theoretic argument shows that $\Omega(n \log \mid \mathcal{U} \mid)$ predicate evaluations (not even necessarily restricted to unary predicates) are required to evaluate the identity function, that is to determine all of the arguments $x_1, \cdots, x_n$ exactly. Other functions, for example $f(X) = x_1 + \cdots + x_n$, whose evaluation require complete knowledge of all arguments, are similarly shown to require $\Omega(n \log \mid \mathcal{U} \mid)$ unary predicate evaluations. It is of interest to identify or characterize functions $f$ that do not have this property, that is for which the exhaustive $O(n \log \mid \mathcal{U} \mid)$ comparison algorithm is not asymptotically optimal.

Hereafter we will restrict our attention to the cases where $f$ is either max or min, the complexity of which are well understood on a binary comparison model [6]. In general we denote the value $\max\{x_1, \cdots, x_n\}$ (respectively $\min\{x_1, \cdots, x_n\}$) by $x_{\max}$ (respectively, $x_{\min}$). The possibility for improvement over the naive $O(n \log \mid \mathcal{U} \mid)$ algorithm suggests itself almost immediately in these cases. Obviously, if one has determined the value of $x_1$ (or more generally $\max\{x_1, \cdots, x_i\}$) then the universe of interest for all subsequent values has been reduced. (If $x_{i+1}$ is determined to be no larger than $\max\{x_1, \cdots, x_i\}$ then it can be ignored without being known explicitly.) However, this observation alone leads to an algorithm using $O(n \log(\mid \mathcal{U} \mid /n + 1))$ predicate evaluations in the worst case, which provides no asymptotic improvement for large universes.

The naive algorithm (and, presumably any other approach that has suggested itself to the reader) exploits a total order on $\mathcal{U}$ and uses only *threshold predicates*, that is questions of the form "is $x_i$ greater than or equal to $c$", for fixed constants $c$. We have intentionally included non-threshold predicates in our formulation of problem since their use is another potential source of improvement over the naive schemes. As it turns out, for the computation of max and min, threshold predicates suffice to realize (to within small constant factors) the lower bounds required even with arbitrary unary predicates.

The problem of computing $\max\{x_1, \cdots, x_n\}$ using $O(n + m)$ threshold predicates, for $\mathcal{U} = \{0, \cdots, 2^m - 1\}$, was originally posed by J. Saxe [4], and solved shortly thereafter by L. Guibas, W. Laaser and J. Saxe. The particular solution presented in this paper (including the slightly more general setting) was derived by D. Kirkpatrick and F. Gao

and was presented in preliminary form in [5].

We begin by (temporarily) restricting our attention to threshold predicates. This serves to simplify the expression of both lower bound arguments and algorithms. Under this restriction our problems can be interpreted as natural matrix searching problems. These matrix problems together with the associated lower bound results are described in section 2. The corresponding algorithms (upper bounds) are presented in section 3. Section 4 interprets the algorithmic results in the communication complexity setting and describes the more general (non-threshold predicates) lower bounds.

Our principal results for the worst-case complexity of computing $x_{\max} = \max\{x_1, \cdots, x_n\}$ can be summarized as follows:

- Determining $x_{\max}$ requires $\Omega(n + \log \mid \mathcal{U} \mid)$ unary predicate evaluations.
- If $\mathcal{U}$ is any set of integers, $x_{\max}$ can be determined in $O(n + \log |x_{\max}|)$ threshold predicate evaluations.
- If $\mathcal{U}$ is the set of positive reals, then $x_{\max}$ can be determined to within absolute error $\epsilon$ in $O(n + \log x_{\max} + \log 1/\epsilon)$ threshold predicate evaluations. If $\mathcal{U}$ is the set of reals in the range $(0, 1]$, then $x_{\max}$ can be determined to within relative error $\epsilon \leq 1$ in $O(n + \log\log 1/x_{\max} + \log 1/\epsilon)$ threshold predicate evaluations.
- The communication complexity of determining $x_{\max}$ in the multiparty broadcast model is $\Omega(n)$ messages and $\Omega(n+\log \mid \mathcal{U} \mid)$ bits and these bounds are simultaneously realizable.

## 2 Lower bounds for threshold predicates

Suppose that $\mathcal{U} = \{0, \cdots, m\}$, where $m \geq 1$. If we restrict our attention to algorithms that use only threshold predicates then the problem of determining $x_{\max} = \max\{x_1, \cdots, x_n\}$ can be recast as a matrix searching problem. Consider the binary matrix $M[1:n, 1:m]$ defined by $M[i,j] = 1$, if $x_i \geq j$ (and $M[i,j] = 0$ otherwise). The evaluation of a threshold predicate on input $x_i$ amounts to a single probe into the matrix $M$. Determining $x_{\max}$ corresponds to locating the rightmost (highest indexed) column of $M$ containing a 1. Since the entries of any row of $M$ are monotonically decreasing, it is clear that exhausive search of $M$ is unnecessary. Indeed, the naive algorithm sketched in the introduction can be viewed as the (binary) search in successive rows for the rightmost one in that row. The question is, is there a significantly better probing strategy? Note that the constraints implicit in the structure of $M$ make the problem fundamentally different from that associated with the detection of graph properties (*cf.* [8]).

We start by formulating a simple adversary-based lower bound on the number of probes required to determine $x_{\max}$.

*Lemma 2.1.* Finding $x_{\max}$ requires $n + \log(m+1) - 1$ probes of $M$ even if it is known that $x_{\max} - 1 \leq x_i \leq x_{\max}$, for $1 \leq i \leq n$.

*Proof.* The adversary responds to successive probes in such a way as to maximize the range of possible values (initially $\{0, \cdots, m\}$) that $x_{\max}$ could assume. When this range has been

reduced to $\{j, j+1\}$, for some $j$, the adversary responds $M[i,j] = 1$ and $M[i,j+1] = 0$ for all subsequent probes. Since each probe reduces the size of the range by at most one half, $\log(m+1) - 1$ probes are required to reduce the range to a size two. Thereafter $n$ probes are required to confirm that $x_{\max} = j$ (and not $j+1$). In fact $2n$ probes are required to identify all $x_i = x_{\max}$. □

*Remark.* Essentially the same strategy shows that $n + \log(m+1) - 1$ probes are required to determine $x_{\min}$ as well.

*Corollary 2.1.* If $\mathcal{U}$ is any finite set of reals then determining $x_{\max}$ or $x_{\min}$ requires $\Omega(n + \log \mid \mathcal{U} \mid)$ threshold predicate evaluations, in the worst case.

*Corollary 2.2.* If $\mathcal{U}$ is the set of integers in the range $\{m_L, \cdots, m_U\}$, where $m_L < m_U$, then determining $x_{\max}$ or $x_{\min}$ requires $\Omega(n + \log(m_U - m_L))$ threshold predicate evaluations, in the worst case.

It is clear from corollary 2.2 that the worst case complexity of finding extrema in an arbitrary collection of integers can be arbitrarily high. A third corollary suggests that it may be more natural to attribute this complexity to the size of the output.

*Corollary 2.3.* If $\mathcal{U}$ is the set of integers in the range $\{2^k + 1, \cdots, 2^{k+1}\}$, where $k \geq 1$, then determining $x_{\max}$ (or $x_{\min}$) requires $\Omega(n + \log x_{\max})$ threshold predicate evaluations, in the worst case.

Despite the assumptions stated in Lemma 2.1 (which amount to significant constraints on the adversary) the lower bounds of this section can be realized (to within small constant factors) by a fairly straightforward algorithm. What the lower bounds suggest is that the algorithm must somehow reduce the range of candidates for $x_{\max}$ without investing too many probes in any particular row of $M$.

# 3 Algorithms using threshold predicates

The previous section presented lower bounds on the number of threshold predicate evaluations required to determine $x_{\max}$ (and $x_{\min}$), when $\mathcal{U} = \{0, \cdots, m\}$. For ease of expression of our basic algorithm, it turns out to be convenient to consider the equivalent problem of determining $x_{\max}$ to within absolute error less than 1, when $\mathcal{U} = [0, m)$ (the set of positive reals less than $m$). (The equivalence is established by noting that i) $\lfloor x_i \rfloor = \max\{\lfloor x_1 \rfloor, \cdots, \lfloor x_n \rfloor\}$ implies $\max\{x_1, \cdots, x_n\} - x_i < 1$, and ii) the reverse implication holds when all of the $x_i$'s are integers.)

In keeping with the earlier matrix formulation, let $M : \{1, \cdots, n\} \times [0, m+1] \rightarrow \{0, 1\}$ be defined by $M[i,z] = 1$ if and only if $x_i \geq z$. As the algorithm proceeds the state of knowledge about $x_i$ is captured entirely by the positions of the rightmost 1 and the leftmost 0 discovered (so far) in row $i$. We denote these quantities by $\rho(i)$ (rightmost 1) and $\lambda(i)$ (leftmost 0), respectively. (Initially, $\rho(i) = 0$ and $\lambda(i) = m+1$). We denote by $\rho_{\max}$ the value $\max\{\rho(i) : 1 \leq i \leq n\}$, the rightmost 1 discovered so far. At any point in time row $i$ is said to be *active* if $\lambda(i) > \rho_{\max}$, that is it remains a possibility that $x_i > \rho_{\max}$.

The idea of the algorithm is to maintain a set $I$ consisting of the indices of all active rows. The algorithm continues until $\lambda(i) \leq \rho_{\max} + 1$, for all $i \in I$, at which point

$x_{\max} - 1 < \rho_{\max} \leq x_{\max}$. More specifically, $I$ is maintained as a collection of disjoint sets $B_s, B_{s+1}, \cdots, B_t$ (called *blocks*), where i) $\lambda(i) = \beta_k$, for all $i \in B_k$ (i.e. indices in the same block share the same $\lambda$-value) and ii) the block sizes and gaps between $\lambda$-values of successive blocks satisfy the relationship $\mid B_k \mid = \log(\beta_{k-1} - \beta_k) - \log(\beta_k - \beta_{k+1})$.

We are now in a position to present the algorithm in pseudo-code. We use the notation $s \nwarrow S$ (respectively $S \nwarrow s$) to denote the operation of removing an (arbitrary) element of set $S$ and assigning it to the variable $s$ (respectively adding the element denoted by $s$ to the set $S$).

```
procedure find_max (M, n, m)
    (* initialize *)
        s ← t ← 0
        B_0 ← {1, ..., n}; β_0 ← m + 1
        ρ_max ← 0
    while β_s > ρ_max + 1 do (* Invariants 1 & 2 *)
        begin
        î ↖ B_s (* select next active row *)
        ĵ ← (ρ_max + β_t)/2 (* select initial probe position *)
        if M[î, ĵ] = 0
            then begin
                (* start a new block *)
                    β_{t+1} ← ĵ
                    B_{t+1} ↖ î
                    t ← t + 1
                    end
            else begin
                (* find highest indexed old block to assign î to *)
                    ρ_max ← ĵ
                    ĵ ← β_t
                    while t > s & M[î, ĵ] = 1 do begin (* Invariants 3 & 4 *)
                        (* eliminate block B_t *)
                            B_t ← φ
                            t ← t − 1
                        ρ_max ← ĵ
                        ĵ ← β_t end
                    if t = s then begin
                        B_{t+1} ← B_t
                        β_{t+1} ← β_t
                        s ← t ← t + 1 end
                (* add î to block B_t *)
                    B_t ↖ î
                end
        if α_s = 0 then s ← s + 1
        end
    return ρ_max
```

The (partial) correctness of the above procedure hinges on two simple observations. The first is the fact (mentioned earlier) that whenever a row index $i$ is eliminated (no longer belongs to $\cup_{k=s}^{t} B_k$), $\lambda(i) \leq \rho_{\max}$ (that is row $i$ is no longer active). The second observation is that at termination $\beta_k \leq \rho_{\max} + 1$, for all $k \geq s$, and consequently $\rho_{\max} > x_{\max} - 1$. The termination (and hence correctness) of the procedure *find_max* is an immediate consequence of the following complexity analysis.

To analyse the complexity of our procedure it suffices to verify some additional invariant properties. Specifically, if we define $\alpha_i = \mid B_i \mid$, for $i \geq 0$, $\gamma_t = (\beta_t - \rho_{\max})/2$, and $\gamma_k = \beta_k - \beta_{k+1}$, for $s \leq k < t$ (see Figure 1 for a snapshot), and if *#probes* denotes the number of probes made at a given point in time, then the following loop invariants are straightforward to verify.

**Outer loop invariants:**

$$\#probes = 2n + 2\log(m+1) - 2\alpha_s - 2\log\gamma_s - t - 2 \tag{1}$$

and

$$\gamma_{k-1} = \gamma_k \cdot 2^{\alpha_k}, \text{ for } s < k \leq t \tag{2}$$

**Inner loop invariants:**

$$\#probes = 2n + 2\log(m+1) - 2\alpha_s - 2\log\gamma_s - t - 3 \tag{3}$$

and

$$\gamma_{k-1} = \gamma_k \cdot 2^{\alpha_k}, \text{ for } s < k < t, \text{ and } \gamma_{t-1} = \gamma_t \cdot 2^{\alpha_t + 1} \tag{4}$$

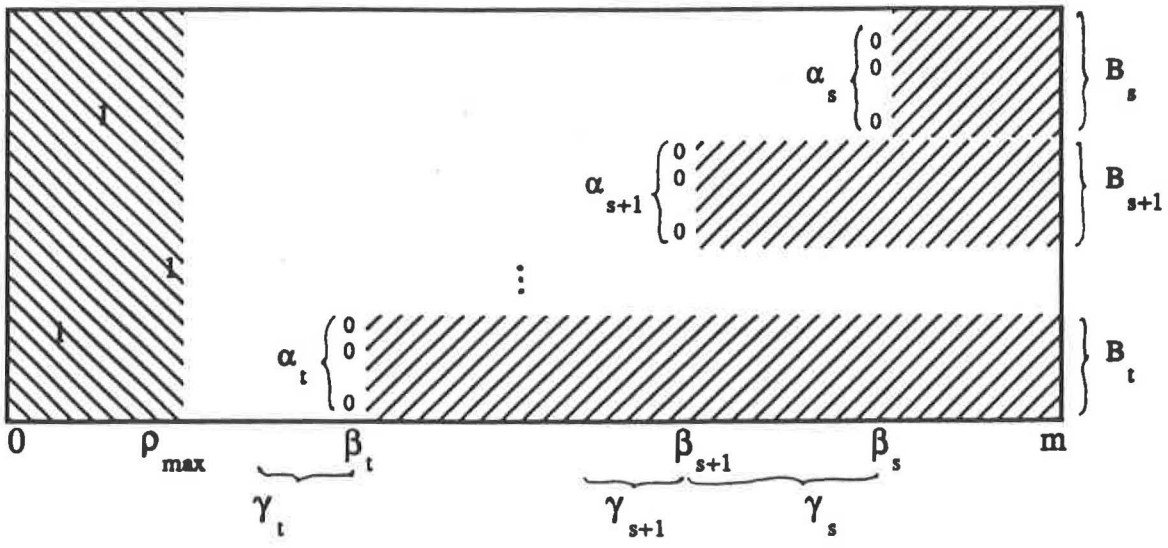


Figure 1

Since $\gamma_s \geq 1$ prior to termination, it follows from the above that $\log\gamma_s + \alpha_s \geq 0$ up to and including the time of termination. Since, in addition, $t \geq 0$ at all times, it follows immediately that *#probes* is bounded above by $2(n + \log(m+1))$. Hence we have established the following:

*Lemma 3.1.* Procedure *find_max* determines $x_{\max}$ to within absolute error less than 1 using at most $2(n + \log(m+1) - 1)$ probes.

We now turn our attention to some direct applications of lemma 3.1.

*Theorem 3.1.* If $\mathcal{U}$ is any set of reals then $x_{\max}$ and $x_{\min}$ can be determined in $2(n + \log|\mathcal{U} - 1)$ threshold predicate evaluations.

*Proof.* Exploit the natural (order preserving) bijection from $\mathcal{U}$ to $\{0, \cdots, |\mathcal{U}| - 1\}$. To compute $x_{\min}$ note that $\min\{x_1, \cdots, x_n\} = m - \max\{m - x_1, \cdots, m - x_n\}$. □

*Theorem 3.2.* If $\mathcal{U}$ is the set of all integers then $x_{\max}$ and $x_{\min}$ can be determined in $O(n+\log(|x_{\max}|+1))$ and $O(n+\log(|x_{\min}|+1))$ threshold predicate evaluations respectively.

*Proof.* It suffices to compute suitable upper bounds on $|x_{\max}|$ and $|x_{\min}|$ efficiently. For example, if $x_{\max}$ is positive the following fragment determines a bound $m$ satisfying $m/2 \leq x_{\max} < m$ in $O(n + \log(x_{\max} + 1))$ probes.

$\hat{i} \leftarrow 1, m \leftarrow 1$
**while** $\hat{i} \leq n$ **do begin**
  **if** $M[\hat{i}, m] = 1$
    **then** $m \leftarrow 2m$
    **else** $\hat{i} \leftarrow \hat{i} + 1$
  **end**

Similarly, assuming $x_{\min}$ is positive the following fragment determines a bound $m$ satisfying $m/2^n \leq x_{\min} < m$ in $O(n + \log(x_{\min} + 1))$ probes.

$\hat{i} \leftarrow 0, m \leftarrow 1$
**while** $M[\hat{i} + 1, m] = 1$ **do begin**
  $m \leftarrow 2m$
  $\hat{i} \leftarrow (\hat{i} + 1) \bmod n$
  **end**

The cases where $x_{\max}$ and $x_{\min}$ are negative are similar. □

*Theorem 3.3.* If $\mathcal{U}$ is the set of all positive reals then $x_{\max}$ (respectively $x_{\min}$) can be determined to within absolute error $\epsilon > 0$ in $O(n + \log x_{\max} + \log 1/\epsilon)$ (respectively $O(n + \log x_{\min} + \log 1/\epsilon)$) threshold predicate evaluations.

*Proof.* It suffices to apply Theorem 3.2 to the set $\{y_1, \cdots, y_n\}$ where $y_i = \lceil x_i/\epsilon \rceil$. □

*Theorem 3.4.* If $\mathcal{U}$ is the set of all positive reals in $(0, 1]$ then $x_{\max}$ (respectively, $x_{\min}$) can be determined to within relative error $\epsilon$, where $0 < \epsilon \leq 1$, in $O(n+\log\log(1/x_{\max})+\log 1/\epsilon)$ (respectively, $O(n + \log\log(1/x_{\min}) + \log 1/\epsilon)$) threshold predicate evaluations.

*Proof.* In this case we apply Theorem 3.3 to the set $\{z_1, \cdots, z_n\}$ where $z_i = \log_{1+\epsilon} 1/x_i$ and observe that an absolute error of 1 in determining $\min\{z_1, \cdots, z_n\}$ corresponds to a relative error of $\epsilon$ in determining $\max\{x_1, \cdots, x_n\}$. Since $\log z_i = \log\log 1/x_i - \log\log 1 + \epsilon$ and $\log 1 + \epsilon \geq \epsilon$ for $\epsilon \in (0, 1]$, the result follows. □

# 4 Communication complexity of extrema finding

As we noted in the introduction, the determination, using unary predicates, of $\max\{x_1, \cdots, x_n\}$ corresponds in a natural way to the distributed evaluation of $\max\{x_1, \cdots, x_n\}$ on a multiparty broadcast model of distributed computation. In fact a slight variant of our algorithm *find_max* presented in section 3 translates to a optimal protocol for the cooperative evaluation of $\max\{x_1, \cdots, x_n\}$. In this protocol processor $P_i$ holds $x_i$ and broadcasts (in its turn) bits from specific positions of row $i$. If we ignore the rows that have ceased to be active (the corresponding processors become passive in the protocol) a natural implementation of procedure *find_max* processes rows of $M$ in a round-robin fashion. When its row is selected as the next active row, processor $P_i$ simulates the sequence of probes of the ith row of $M$ specified by the procedure *find_max*, and packages the corresponding bit sequence into a single message for broadcast to the other processors. This information suffices for all of the other processors to update their ongoing simulations of procedure *find_max*.

As it has been described, each probe made by procedure *find_max* corresponds to a single bit of communication in the protocol. Thus Theorem 3.1 can be reinterpreted as asserting that $x_{\max}$ and $x_{\min}$ can be computed using $O(n + \log \mid \mathcal{U} \mid)$ bits of (broadcast) communication in a distributed computation model. Note that the protocol may use asymptotically as many messages as it does bits (since the procedure *find_max* may make only $O(1)$ probes for each repetition of its outer loop). However a very simple modification of procedure *find_max* ensures that the corresponding communication protocol uses only $O(n)$ messages while retaining the same asymptotic bound on the number of bits. The idea is to replicate rows $\lceil(\log \mid \mathcal{U} \mid)/n\rceil$ times. The resulting matrix has $n' = O(n + \log \mid \mathcal{U} \mid)$ rows (with, of course, the same value of $x_{\max}$) and hence requires (asymptotically) the same number of probes. Assuming that bits from replicated rows are bundled into the same message, it follows that each message contains at least $\lceil(\log \mid \mathcal{U} \mid)/n\rceil$ bits and hence there are $O(n)$ messages in total. We summarize the above construction in the following:

*Theorem 4.1.* There exists a multiparty communication protocol that determines $\max\{x_1, \cdots, x_n\}$ in $O(n)$ (broadcast) messages with a total of $O(n + \log \mid \mathcal{U} \mid)$ bits.

We conclude this section by outlining our argument that the protocol described in Theorem 4.1 is asymptotically optimal. Specifically:

*Theorem 4.2.* Any communication protocol that determines $\max\{x_1, \cdots, x_n\}$ in the multiparty broadcast model requires at least $n$ messages and $n + \log|\mathcal{U}| - 1$ bits.

*Proof.* The message lower bound is clear; in the worst case each participant must communicate something (here we exploit asynchrony) or else the value of $\max\{x_1, \cdots, x_n\}$ can not be known to all processors.

The bit lower bound requires a modest generalization of Lemma 2.1. First, we note that our translation of an algorithm for determining $x_{\max}$ with unary predicates to a (broadcast) protocol for computing $x_{\max}$, can be reversed. If we are only interest in bit complexity any protocol can be trivially modified to use only 1-bit messages. Since each such message can be interpreted as the evaluation of some unary predicate on one of the inputs, a protocol specifies an algorithm for determining $x_{\max}$ using unary (but not necessarily threshold) predicates.

At any point in time there is partial information available to all of the processors about each of the numbers $x_i$. If a specific number is consistent with this partial information we say that it is an *i-candidate*. The adversary maintains the subset $\mathcal{C}$ of $\mathcal{U}$ consisting of numbers that are $i$-candidates, for all $i$. With each successive unary predicate the adversary responds in such a way that the size of $\mathcal{C}$ is decreased by at most 2. Since $\mathcal{C} = \mathcal{U}$ initially, after $\log |\mathcal{U}| - 1$ predicate evaluations $|\mathcal{C}| \geq 2$. Thereafter, if each processor holds $\min\{x | x \in \mathcal{C}\}$, at least one additional bit must be broadcast by each processor to confirm this fact. □

*Corollary 4.1.* If $\mathcal{U}$ is any set of reals then determining $x_{\max}$ (or $x_{\min}$) requires $\Omega(n + \log |\mathcal{U}|)$ unary predicate evaluations, in the worst case.

# 5 Concluding remarks

We have identified the complexity of extrema finding using unary predicates to within a factor of 2. It will be clear that the upper bound can be improved slightly by simply avoiding questions (probes) whose results are implied by earlier questions. (In fact, a slightly more careful analysis of our algorithm shows that it achieves the optimal bound of $\log(m+1)$ predicate evaluations, when $n = 1$.) It remains to specify exactly the complexity of extrema finding in this model, when $n \geq 2$.

It is of interest to study the complexity of other basic functions (for example, finding the two largest elements, or finding the median) in this same model. Similarly, our results motivate the investigation of the multiparty communication complexity of other basic functions.

Our model ignores the cost of choosing and evaluating appropriate threshold predicates for a given universe $\mathcal{U}$. This is reasonable when the total order on $\mathcal{U}$ is given explicitly (for example, by means of some easily computed index function). However, when this is not the case the full complexity of extrema finding is not necessarily captured by the number of unary predicate evaluations required. To illustrate this issue and the role our results can play in its resolution, we point out the following result which is a direct consequence of Theorem 3.3 and the central result of [7].

*Theorem 5.1.* If $\mathcal{U}$ is the set of all rationals with numerator and denominator in $\{1, \cdots, m\}$, then $x_{\max}$ can be determined in $O(n + \log m)$ threshold predicate evaluations and $O(n + \log m)$ arithmetic operations on integers of size at most $2m$.

# References


[1] Aho, A.V., Hopcroft, J.E. and Ullman, J.D., *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1975.

[2] Babai, L., Nisan, N. and Szegedy, M., Multiparty protocols and logspace-hard pseudorandom sequences, *Proc. 21st ACM Symposium on theory of Computing* (1989), 1-11.

[3] Chandra, A., Furst, M. and Lipton, R., Multiparty protocols, *Proc. 15th ACM Symposium on Theory of Computing* (1983), 94-99.

[4] Guibas, L.J.(ed.), Problems, *Journal of Algorithms* 1(1980), 209-212.

[5] Kirkpatrick, D.G. and Gao, F., Finding extrema with unary predicates, *Proc. SIGAL International Symposium on Algorithms* (August 1990), 400-413.

[6] Knuth, D.E., *The Art of Computer Programming, Vol.3*, Addison-Wesley, 1973.

[7] Papadimitriou, C.H., Efficient search for rationals, *Information Processing Letters* 8,1 (Jan. 1979), 1-4

[8] Rivest, R. and Vuillemin, J., On recognizing graph properties from adjacency matrices, *Theoretical Computer Science* 3 (1978), 371-384.

[9] Yao, A.C., Some complexity questions related to distributed computing, *Proc. 11th ACM Symposium on Theory of Computing* (1979), 209-213.