GUESSING GAMES AND DISTRIBUTED COMPUTATIONS IN SYNCHRONOUS NETWORKS

(draft)

J.E. van Leeuwen[1], N. Santoro[2], J. Urrutia[3], S. Zaks[4]

[1]Department of Computer Science, University of Utrecht, Utrecht, 3508 TA, The Netherlands

[2]School of Computer Science, Carleton University, Ottawa, K1S 5B6, Canada

[3]Department of Computer Science, University of Ottawa, Ottawa, K1N 5B4, Canada

[4]Department of Computer Science, Technion, Haifa, 3200, Israel

1. INTRODUCTION

1.1 Background and Motivations

A ***distributed system*** is a network G=(V,E) of |V|=**n** processors connected by |E|=**e** direct communication links, where each processor has a local non-shared memory and can communicate by sending messages to and receiving messages from its neighbours. The behaviour of these processors can be conveniently described as *finite-state* and *event-driven*; that is, each processor at any time is in a particular state and, when a predefined event occurs (e.g., a message is received, the internal clock reaches a predetermined value, etc.), it will serially perform some operations whose nature depends on the current state and on the occurred event. The operations that can be performed are local computations, transmission of messages, and change of state. A ***distributed algorithm*** is the specification of what operations must be serially performed by a processor when a predefined event occurs in a given state. To ensure a fully distributed computation in the system, it is assumed that every processor has the same algorithm.

A fundamental computation in this environment is the ***election*** process; that is, the process of changing from an initial system configuration, where every processor in the network is in the same state, to a final configuration where exactly one processor is in a predefined state (say, *elected*) and all other processors are in another predefined state (say, *defeated*). Note that there is no a-priori restriction on which processor should become *elected*.

A related basic computation is the ***minimum-finding*** process; that is, the process of changing from an initial system configuration where every processor has an associated value (from a totally ordered set) and is in the same state, to a final configuration where the processors with the smallest value are in a predefined state (say, *minimum*) and all other processor are in another predefined state (say, *large*).

Another related computation is the ***spanning-tree construction*** process; that is, the process of determining at each processor a subset of its incident links such that the overall collection of these subsets form a spanning-tree of the original network.

If each processor has a *distinct* value (e.g., identity), the election problem can obviously be solved by first determining the smallest value and then electing the processor which has this value; furthermore, once a leader is elected, a spanning tree can be easily constructed using a distributed (single-initiator) graph traversal algorithm (e.g., see [7]) with an additional O(**e**) messages. Assuming distinct identities, different (upper and lower) bounds for all three problems have been established depending on the actual topology of the network and on the amount of topological information available to the network processors. For example, it has been shown that Ω(**n** log **n**) messages need to be transmitted in a *ring* of **n** processors ([6],[9],[19]); several algorithms achieving this bound have been presented (e.g., [5,6,8,15,20]). In the case of *complete* networks, the number of messages needed to elect a leader has been shown to be O(**n**) and O(**n** log **n**) depending on whether the processors have available a global sense of orientation [16] or not [14], respectively. In the case when the network topology is unknown (the *arbitrary graph* case), an election can be performed exchanging O(**e**+**n** log **n**) messages [12] which is known to be worst-case optimal (e.g., see [2,21]). It should be noted that in all the above solutions, a message is allowed to contain a processor value; hence, the total number of *bits* transmitted is equal to the number of messages multiplied by O(log **i**), where **i** is the smallest value in the network. Probabilistic algorithms which exchange fewer bits in *rings* have been recently presented [1]. All these bounds have been derived without any assumption on the network *synchronicity* .

In a *synchronous* network each processor has a clock locally available, and all clocks "tick" simultaneously (although they might not all sign the same absolute time); furthermore, if a processor sends a message at local time t to a neighbour, the message is received and processed there at time t+1 (sender's time). By exploiting synchronicity, Frederickson & Lynch [9] and Vitanyi [23] independently established that a leader can be elected in a *ring* using O(**n**) messages; since a message carries a processor value, the total number of *bits* transmitted by the algorithm is O(**n** log **i**), where **i** is the smallest value in the network. However, the number of synchronous rounds (i.e., time units) required by their algorithm is O(**n** $2^{\mathbf{i}}$) which is exponential in the range of the values. The time bound has been subsequently reduced to O(**n** $2^{\mathbf{n}}$ + $\mathbf{i}^2$) with the same message and bit complexity by Gafni [11]; a further reduction in time to O(**n** **i**) has been obtained by Marchetti [17] at the expense of a O(**n** log **n** log **i**) bit complexity. A more recent result, by Overmars and Santoro [18], shows a time *vs* bit trade-off which, at one end of the spectrum, yields the bounds of O(**n** log **n**) bits and O(**n** **i**) time; thus, it makes the bit complexity independent of the processor values, and the time polynomial in both **n** and **i**. All the above results have been established without any information about **n** at the processors. If the network size **n** is known to the processors, Santoro and Rotem [22] showed that O(**e**) bits and O(**n** **i**) time suffice to elect a leader in an *arbitrary network* (a similiar result was established for *rings* by Gafni [11]).

If the assumption on the uniqueness of the values does not hold (the ***anonymous*** network case), the election problem cannot obviously be solved by an extrema-finding process. Furthermore, Angluin [2] has shown that, if the processors have no values (or, analogously, all have the same value) then no deterministic solution exists for the election problem, duly renamed ***symmetry breaking problem***, regardless of whether the network is synchronous or not. (Probabilistic solutions to this version of the problem are however possible if both symmetry and knowledge of $\mathbf{n}$ are assumed [10,13,22].) In spite of this negative characterization for the election problem, certain computations can still be deterministically performed in synchronous anonymous networks provided that some information about the network is available to the processors; for example, if each value is a bit and the network is a *ring* whose size $\mathbf{n}$ is known to all processors, then the functions *sum* and *xor* can be computed in $O(\mathbf{n}^2)$ bits and $O(\mathbf{n})$ time, while the functions *and* and *or* can be computed in $O(\mathbf{n})$ bits and $O(\mathbf{n})$ time [3]. In particular, knowing the number $\mathbf{n}$ of processors, the function *min* (that is, the minimum-finding process) can be computed using $O(\mathbf{e})$ bits in time $O(\mathbf{n}\ \mathbf{i})$ in an arbitrary network [22].

1.2 Main Results

In this paper, it is shown that in an *anonymous synchronous network* where $\mathbf{n}$ is known the smallest value can be determined in $O(k\ \mathbf{e})$ bits and $O(k\ \mathbf{i}^{1/k}\ \mathbf{n})$ time for any integer $k>0$. The choice of k offers a time *vs* communication trade-off; in particular, by choosing a fixed k, the existing $O(\mathbf{i}\ \mathbf{n})$ time bound is reduced with still a linear number of bits. Furthermore, it is shown that this trade-off is optimal for the class of solutions considered here.

As a consequence, election and spanning-tree construction in a*synchronous networks with unique values* can both be performed in $O(k\ \mathbf{e})$ bits and $O(k\ \mathbf{i}^{1/k}\ \mathbf{n})$ time for any $k>0$, improving the existing $O(\mathbf{e})$ bits and $O(\mathbf{n}\ \mathbf{i})$ time bounds.

These results apply *regardless of the network topology* to both *undirected* networks (i.e., where the communication links are bidirectional) and *strongly-connected directed* networks (i.e., where the communication links are unidirectional but it is possible to route a message from each processor to any other processor). For brevity, the results will be presented here only for unidirected networks.

Beside the improved bounds, an interesting contribution of this paper consists in the reformulation of the minimum-finding problem in terms of a number-guessing game, independent of the network topology and of its communication capabilities. The upper-bounds are obtained by simply developing a solution for this game; the optimality is proved by studying a related guessing game, determining the unique solution, obtaining a closed formula for the complexity of this solution, and reinterpreting this result in terms of the original game.

It is also shown that the results for these games lead to improved solutions for the ***variable-cost comparison searching*** problems studied by Bentley and Brown [4]; this, in turn, offers an improvement in some of the applications where the solutions by Bentley and Brown had been applied: sensitivity analysis, broadcasting a point on a line, linear recursion.

Finally, the proposed solution for minimum-finding leads to a new algorith for *symmetry breaking* for the class of networks considered in [10,22] and whose complexity matches the one of the most efficient algorithm for this class presented in the literature [10].

2. MINIMUM FINDING AS A GUESSING GAME

A *distributed system* can be described as a couple G=(V,E) where V is a set of processors and E is a set of communication links between processors; if (u,v)E, processors u and v are said to be *neighbours*; each processor u has an associated value i_u (e.g., an identity) from a totally ordered set in its local non-shared memory, and communicates by sending messages to its neighbours. If the values i_u are not (known to be) distinct, the system is said to be *anonymous*. In a *synchronous* distributed system G, each processor u has locally availble a clock c_u; all clocks "tick" simultaneously (although they might not all sign the same absolute time), and a message sent by a processor at local time t to a neighbour is received and processed there at time t+1 (sender's time). At any time, a processor is in a *state* (from a finite predefined set).

The *minimum-finding* problem is the problem of distributively computing **i**=Min$\{i_u\}$; i.e., determining the smallest of the associated values. It is assumed that, initially, all processors are in the same state, know **n**, and simultaneously start the execution of the minimum-finding computation; at the end of the computation, all processors whose associated value is **i**=Min$\{i_u\}$ must be in a predefined state, all others must be in another predetermined state.

In this section, a class of solution algorithms is characterized by reformulating the minimum-finding problem in terms of a number-guessing game.

Consider the following distributed algorithm, where x is a parameter available to all processors:

```
DECIDE(x)
1.    Set clock:=0 and Senders:=Ø. If local_value≤x send "YES" to all neighbours
       and become decided, otherwise become undecided. Start counting.
2.    If a "YES" is received with clock<n then:
      if undecided: add to Senders the neighbour from which the "YES" has been
      received, send the message to the neighbours not in Sender, and become
      decided; otherwise, ignore the message.
```

An immediate and important property of this algorithm is expressed by the following lemma:

Lemma 1 Let all processors know **n** and x, and simultaneously start the execution of DECIDE(x). Then, at time **n**:

(i) if all local values are greater than x, all processors have become *undecided*;

(ii) if there is at least one local value $i \leq x$, all processors are *decided.*

Furthermore, the number of *bits* transmitted is zero in case (i) and at most 2**e** in case (ii).

Proof. (i) A processor will become *undecided* and send no message iff its value is greater than x (step 1 of DECIDE). Thus, if all values are greater than x, no messages will be transmitted during the execution of DECIDE; furthermore, all processors will remain *undecided.*

(ii) If a processor v has a value $i \leq x$, it will become decided, and send a YES to all its neighbours (step 1 of DECIDE). Upon receiving such a message, an *undecided* processor becones *decided* and forwards the message to all of its neighbours (step 2 of DECIDE). A decided processorignores all received messages, while an *undecided* processor forwards a received YES only to the neighbours from which such a message has not yet been received; thus, at most two messages will be transmitted on each edge for a total of at most 2e bits. Since an *undecided* processor becomes *decided* as soon as it receives a message, all processors will become decided within n-1 time units.

Using this property, a technique for minimum-finding can be developed by performing a sequence of executions of DECIDE as follows. Initially, all nodes choose the same initial value x_1 and simultaneously performe DECIDE(x_1). After **n** time units, all nodes will be aware of whether the minimum value is greater or not than x_1 (situation (i) and (ii), respectively); based on this outcome, a new value x_2 will be chosen by all nodes which will then simultaneously perform DECIDE(x_2). In general, based on the outcome of the execution of DECIDE(x_i), all nodes will choose a value x_{i+1} and simultaneosly perform DECIDE(x_{i+1}); the process is repeated until the minimum value is unambigously determined. Depending on which strategy is employed for choosing x_{i+1} given the outcome of DECIDE(x_i), different minimum-finding algorithms will result from this technique.

This technique allows to reformulate the minimum-finding problem in terms of a *number-guessing game* as follows:

Guessing Game

1. the network is a player;
2. the minimum value in the network is a number, previously chosen and unknown to the player, which must be guessed;
3. the player can ask questions of the type "is the number greater than x?", where each question corresponds to a simultaneous execution of DECIDE(x);
4. situations (i) and (ii) of Lemma 1 correspond to a "yes" and a "no" answer to the question, respectively.

First observe that, by definition, to each solution strategy for the number-guessing game corresponds a solution algorithm for the minimum-finding problem. As for the complexity of these solution algorithms, recall that, by Lemma 1, each execution of DECIDE (i.e., each question) requires **n** time units; on the other hand, the number of bits transmitted is either zero or at most 2**e**, depending on whether the answer is "yes" or "no", respectively. In terms of the game, this situation can be interpreted as if there exist two charging accounts, *T(ime)* and *B(it)*: the player is charged one unit in the *T* account for each question asked, and one unit in the *B* account for each ***over-estimate*** (i.e., each question where the answer is "no"). Therefore, a strategy which allows the player to guess the number using t units in the *T* account and b units in the *B* account corresponds to a minimum-finding algorithm which requires **n**t time and at most 2**e**b bits; that is

Theorem 1 Let S be a solution strategy for the number-guessing game which requires b(x) overestimates and a total of t(x) questions in the worst case, where x is the unknown number. Then:

i) minimum-finding can be performed in an anonymous synchronous network using at most **n** t(**i**) time and 2 **e** b(**i**) bits;

ii) election in a synchronous network with distinct values can be performed using at most **n** t(**i**) time and 2 **e** b(**i**) bits;

iii) a spanning-tree can be constructed in a synchronous network with distinct values using at most **n** t(**i**) time and 2**e**(b(**i**) +2) bits;

where **i** is the smallest value in the network, provided **n** is known.

3. GUESSING GAMES

In this section, some variations of the game described in the previous section are considered and solved; because of the correspondence between number-guessing game and minimum-finding, these solutions will provide upper and lower bounds on the communication complexity of the distributed problem. All these games will be characterized by the triple <**N**,**t**,**b**>, where **N** denotes the size of the interval in which the number to be guessed is known to lie, **t** is the number of total questions allowed, and **b** is the number of allowed overestimates. In the following, replacing one of these three parameter by the symbol * will indicate that the parameter is unknown or that the goal of the game is to optimize the quantity represented by the parameter.

The games being considered are the following:

1. **<*,t,b>-game**. The unknown number is a positive integer, and the total amounts **t** and **b** of admissible charges are predetermined. The game consists in determining and searching the largest interval [1,N] for which it is always possible to find the unknown number using at most **t** questions and **b** overestimates, given that the unknown number is in the interval.

2. **<N,*,b>-game**. The unknown number is a positive integer in the interval [1,**N**], and the total amount **b** of admissible overestimates is predetermined. The game consists in determining the unknown number using the minimum number of questions. (This game is exactly the *bounded-searching problem with variable-cost comparisons* studied in [4])

3. **<*,*,b>-game**. The unknown number is a positive integer and the total amount **b** of admissible overestimates is predetermined. The game consists in determining the unknown number using the minimum number of questions. (This game is exactly the *unbounded-searching problem with variable-cost comparisons* studied in [4]).

3.1 <*,t,b>-Games and Binomial-Sum Trees

Consider first the case where the unknown number is a positive integer, and the total amounts **t** and **b** of admissible charges are predetermined. The game, called a ***<*, t, b>-game***, consists in:

1. determining the largest interval [1, f(**t**,**b**)] for which it is always possible to find such a number using at most **t** questions and **b** overestimates given that the unknown integer is in the interval; and
2. searching in such an interval using at most **t** questions and **b** overestimates.

In this section, the size f(**t**,**b**) of the largest interval in a <*,**t**,**b**>-game is determined, and the optimal searching strategy is presented. The value f(**t**,**b**) will provide lower-bounds for the other games. Let BIN(x,y) denote the binomial coefficient.

Theorem 2 f(**t**,**b**) = $\sum_{i=0,b'}$ BIN(**t**,i), where b'=Min{**b**,**t**} .

Proof.Let **t** and □ be the number of questions and of overestimates, respectively, allowed in the game, and let Q(x)= "is the number greater than x?" be the first question asked. If the answer is "yes", the unknown number is greater than x and the player is left with **t**-1 questions and **b** overestimates; thus, using Q(x) as the first question and assuming a "yes" answer, the largest interval that can be correctly searched is [1,x+f(**t**-1,**b**)]. If the answer is "no", the unknown number lies in the interval [1,x] and the player is left with **t**-1 questions and **b**-1 overestimates; that is, the largest value of x which allow for a correct solution is x=f(**t**-1,**b**-1). Thus, f(**t**,**b**)=f(**t**-1,**b**-1)+f(**t**-1,**b**) where, for every **t**, f(**t**,1)=**t**+1; since the number of overestimates cannot exceed the number of questions, f(**t**,**t**+j)=f(**t**,**t**) for any j>0. By solving this recurrence relation (e.g., determining the generating function F(x,y) = 1/(1-z)(1-y-z*y)), the theorem follows. ®

In order to determine the solution strategy, first reinterpret the recurrence relation f(**t**,**b**)=f(**t**-1,**b**-1)+f(**t**-1,**b**) with boundary condition f(**t**,1)=**t**+1 as defining a class of binary trees, herein called *Binomial-Sum (or BS) trees*, as follows: a [0,0]-BS tree consists precisely of one external node; a [**t**,**b**]-BS tree for **t≥b** is a binary tree consisting of one internal node whose left subtree is a [**t**-1,**b**-1]-BS tree and whose right subtree is a [**t**-1,**b**']-BS tree, where **b**'=Min{**t**-1,**b**} .

Define a [**t**,**b**]-BS *decision tree* to be a [**t**,**b**]-BS tree in which the leaves have been numbered from left to right from 1 to f(**t**,**b**), where each internal node contains the largest integer in the left subtree, and where the left branches are labeled "≤" while the right branches are labeled ">". An example of [t,b]-BS decision tree is shown in Figure 1.

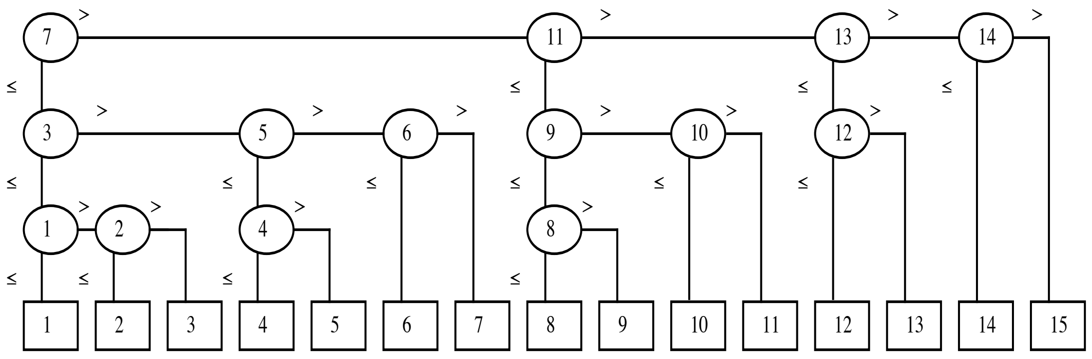

FIGURE 1 The [4,3]-BS decision tree solving the <*,4,3>-game

To determine the unknown number in a <*,**t**,**b**>-game is equivalent to searching in a [**t**,**b**]-BS decision tree: the first question asked in the <*,**t**,**b**>-game will be Q(x)='is the unknown number greater than x?', where x is the value contained at the root of the [**t**,**b**]-BS decision tree; if the answer is "yes" ("no") the next value to be used will be the one contained at the root of the right (left) subtree; and so on, until the leaf containing the desired number is reached (which will be always the case since the unkown number is by assumption in the interval [1,f(**t**,**b**)]).

In a [**t**,**b**]-BS decision tree, the number of internal nodes nodes in a path from the root to any leaf has length at most **t**, and the number of left branches in any such path is at most **b**; thus, the unknown number will always be found using at most **t** questions of which at most **b** are overestimates.

Therefore, the *optimal* solution strategy for a <*,**t**,**b**>-game is as follows:

Optimal Solution Strategy
input:: **t**, **b**; *output*:: X (the unknown number)

```
t:=t; b:=b; p:=0; finished:=false;
while not finished
        x:=p+f(t-1,b-1);
        if answer to Q(x) is "no" then
              if b=1 then
                    X:=x; finished:=true;
              else
                    t:=t-1; b:=b-1;
              endif
        else   /* answer to Q(x) is "yes" */
              if b=1 then
                    X:=x+1; finished:=true;
              else
                    t:=t-1; b:=Min{b,t} ; p:=x;
              endif
        endif
endwhile.
```

The BS decision tree achieving the optimal solution strategy for a <*,4,3>-game is shown in Fig. 1.

An interesting by-product of the above result is the following. Given a sorted array of size N, any element can obviously be found using binary search in at most $\lceil \log N \rceil$ comparisons; however, the number of overestimates is $\lceil \log N \rceil$ -1 in the worst case. However, using the proposed stategy, it is possible to bound the number of overestimates to half of the comparisons using at most one more comparison:

Corollary It is possible to search in a sorted array of size N with $\lceil \log N \rceil$+1 comparisons using at most ($\lceil \log N \rceil$+1)/2 overestimates.

□ □□□□. □($\lceil \log N \rceil$+1, ($\lceil \log N \rceil$+1)/2)) = f($\lceil \log N \rceil$+1, $\lceil \log N \rceil$+1) / 2 = $2^{\lceil \log N \rceil} \geq$ N.

□

3.2 <N,*,b>-Games

Consider the case when the unknown number is a positive integer in the interval [1,**N**], and the total amounts **b** of admissible overestimates is predetermined. The ***<N,*,b>-game*** consists in determining the unknown number asking as few questions as possible.

A solution strategy to this game can be easily obtained using the solution to the previous game Let h(**N**,**b**) denote the smallest integer t satisfying the relation $\sum_{j=0,\mathbf{b}}$ BIN(t,j)≥**N**.

Theorem 3 A <**N**,*,**b**>-game can be solved using at most h(**N**,**b**) questions. Furthermore, this solution is worst-case optimal.

Proof: Let t=h(**N,b**). By employing the solution to the <*,t,**b**>-game (described in the previous section), the interval to be searched will be exactly f(t,**b**); since f(t,**b**)≥**N**, this interval will contain the unknown number. Furthermore, the unknown number will be found asking at most t questions of which **b** are overestimates. Worse case optimality follows from observing that (by Theorem 2) h(**N**,**b**) questions need to be asked when f(t,**b**)=N.

The exact value of h(**N**,**b**) is not known; however, it can be closely bounded as follows:

Lemma 2: $(b!\ N/2)^{1/b}$ + □b/2□-1 < h(N,b) < $(b!\ N)^{1/n}$ +b-1.

Proof: Let t be the smallest value such that f(t,b)≥n. The upper bound follows from N > f(t-1), b) = □$_{j=0,\ b}$ BIN(t-1, j) ≥ BIN(t-1, b) + Bin(t, b-1) = Bin(t, b) ≥ $(t\text{-}b+1)^b$/b!. The lower bound follows from f(t, b) = □$_{j=0,\ b}$ BIN(t, j) < 2 BIN(t,b) < 2 (t+ □b/2 □+ 1)^b / b!.

3.3 <*,*,b>-Games

Consider the case when the unknown number is a positive integer and the total amounts **b** of admissible overestimates is predetermined. The game, called a ***<*,*,b>-game***, consists in determining the unknown number using as few questions as possible.

The main difference betwee this game and the previous ones is that there is no a-priori bound on the value of the unknown number **x** to be guessed. Thus, a solution strategy could be to first determine an interval (1,N) in which **x** lies, and then solve the corresponding <N,*,b'>-game with the remaining b' overestimates.

To bound the value **x**, we will proceed through a sequence of questions Q(g(1)), Q(g(2)),..., Q(g(k)), ..., where g:N->Z is monotonically increasing, until it is determined that g(**j**)≥**x**>g(**j**-1); this will require exactly **j** questions and one overestimate. We are now left to determine **x** in an interval of size Δ(**j**)=g(**j**)-g(**j**-1) with only **b**-1 overestimates; this is exactly a <Δ(**j**),*,**b**-1>-game (described in section 3.2) which can be solved with at most **t'** questions, where **t'**=h(Δ(**j**),**b**-1). Thus, the entire process will require at most **j+t'** questions. Depending on the choice of the function g, different bounds can be obtained.

Theorem 4 A <*,*,**b**>-game can be solved using at most 2t-1 questions, where **x** is the unknown number and t=h(**x**,**b**).

Proof: Choose g(k)=f(k,b). Let t=h(x,b) (i.e. the smallest integer such that f(t,b)≥x); then obviously, j=t. From this follows that □(j)=g(t)-g(t-1)=f(t,b)-f(t-1,b)=f(t-1,b-1); that is, t-1 questions will suffice to solve the resulting <*, □(j), b-1>-game for a total of 2t-1 questions.

4. IMPROVED BOUNDS FOR DISTRIBUTED PROBLEMS

Using the correspondence between guessing games and minimum-finding, the results of the previous section will now be reinterpreted in the context of distributed computations. First observe that, in the distributed problem, no upper-bound is assumed on the range of the values among which the minimum must be found (as in the case of the <*,*,b>-game). Further observe that each solution strategy for the <*,*,b>-game will correspond to a minimum-finding algorithm requiring the transmission of O(b **e**) bits (Theorem 1). Let $\boldsymbol{C}_b$ denote the class of such minimum-finding algorithms.

Theorem 5 The minimum value **i** in a synchronous *anonymous* network can be determined using at most O(k **e**) bits in time O(k **n** $\mathbf{i}^{1/k}$) for any integer k, provided **n** is known and the processors start simultaneously. This bound is optimal among all algorithms in $\boldsymbol{C}_k$ for every value of the integer k.

Proof: Let t be the smalles integer such that f(t. k) ≥ i; by Theorem 4, it follows that i can be guessed using at most 2t-1 questions. Thus, by Theorem 1, the minimum value i can be determined using at most (4t-2)n time and 2ke bits. By Lemma 2, t < $(k!\ i)^{1/k}$ + k –1 which is approximately $i^{1/k}$ k/e + k –1 (using Stirling’s approximation), from which the bound follows. Bt Theorem 2 and Lemma 2, any given algorithm in C_k requires at least t > $(k!\ i/2)^{1/k}$ questions from which optimality follows.

The choice of k in the above theorem yields a *bit vs time* trade-off. In particular, by choosing k=O(1), the existing O(**n i**) time bound for minimum finding in an anonymous network where **n** is known and the processors start simultaneously [22] is improved without increasing the order of magnitude of the bit complexity.

In a similiar way, the following theorem can be proved

Theorem 6 In a synchronous network *with distinct values* election and spanning-tree construction can be performed using at most O(k **e**) bits in time $O(\mathbf{n}\ \mathbf{i}^{1/k})$ for any k, where **i** is the smalles value in the network, provided **n** is known and the processors start simultaneously.

Again, by choosing k=O(1), the theorem yields an improvement in the time complexity without increasing the order of magnitude of the bit complexity.

5. APPLICATIONS

5.1 Searching with Variable-Cost Comparisons

In their study of resource trade-offs, Bentley and Brown [4] have examined problems related or reducible to searching when the cost of performing a comparison depends on the outcome of the comparison itself; namely, they assume that there are two diffent cost measures, C_1 and C_2, and that a comparison (e.g., of x and y) which will charged a C_1 or C_2 cost unit depending on whether $x>y$ or $x\leq y$, respectively. It is not difficult to see that this is exactly the situation described by the guessing game described in section 2. In particular, they cosider and propose solutions for two problems, bounded-searching and unbounded searching, occurring in such an environment. In the ***bounded searching*** problem, the total number of c_2 charges is predetermined, as well as an interval [1,N] where the search must take place; the problem is to devise a strategy which allows to search in the interval using no more that the allowed number of C_2 charges with as few number of C_1 charges as possible. In the ***unbounded searching*** problem, the total number of C_2 charges is again predetermined but no upper-bound on the size of the interval to be searched is known; the problem is to devise a strategy which allows to search using no more that the allowed number of C_2 charges and with as few number of C_1 charges as possible. Let b denote the number of allowed C_2 charges; then the bounded searching problem is exactly the <N,*,b>-game described in section 3.2, and the unbounded searching problem is exactly the <*,*,b>-game described in section 3.3.

Their solutions to these two problems are based on ***binomial decision trees*** (for a detailed description of these trees, the reader is referred to [4]), much in the same way the solutions proposed in this paper can be seen as based on BS decision trees (see section 3.1). Furthermore, their technique can also be used to solve the <*,t,b>-game. In the rest of this section, the solutions described in Section 3 are compared with (and shown to improve upon) the solutions obtained using Bentley and Brown's technique.

Using the searching process for a (**t**+1,**b**)-binomial decision tree as a solution strategy for the <*,**t**,**b**>-game provides an efficient (but not optimal) solution to that problem; in fact it yields an interval of size BIN(**t**+1,Min{**b**,$\lfloor$(**t**+1)/2$\rfloor$}), where BIN(x,y) denotes the binomial coefficient. The solution to the <*,4,3>-game generated by the (5,3)-binomial decision tree is shown in Figure 2; the size of the interval achieved with this solution should be contrasted with the one obtained with the optimal solution stategy presented in section 3.1 and shown in Figure 1.

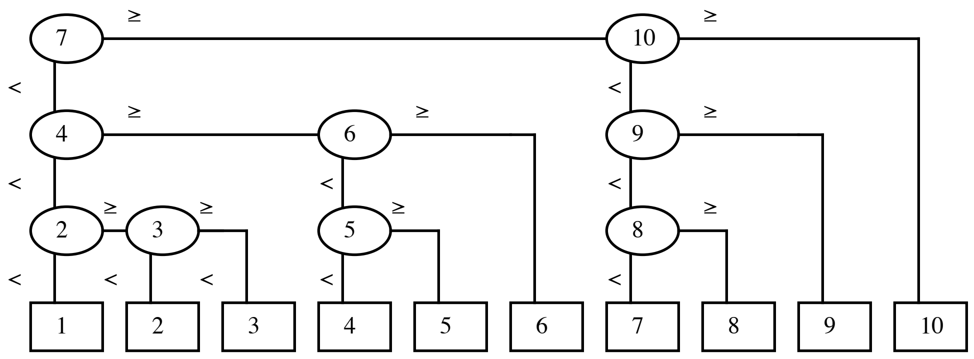

FIGURE 2 The (5,3)-binomial decision tree solving the <*,4,3>-game

The difference between the size of the interval obtained by using binomial trees and the size in the optimal solution (using BS trees) is exactly

$$f(\mathbf{t},\mathbf{b})-\mathrm{BIN}(\mathbf{t}+1,\mathbf{b})= \Sigma_{j=0,\mathbf{b}}\,\mathrm{BIN}(\mathbf{t},j) -\mathrm{BIN}(\mathbf{t}+1,\mathbf{b}) = \Sigma_{j=0,\mathbf{b}-2}\,\mathrm{BIN}(\mathbf{t},j) = f(\mathbf{t},\mathbf{b}-2)$$

which is greater than zero whenever **b**>1; in the case **b**=1, the two solutions coincide.

The <**N**,*,**b**>-game discussed in section 3.2 is exactly the *bounded-searching problem with variable-cost comparisons* studied by Bentley and Brown [4]; their solution requires μ questions (using a (μ+1,**b**)-binomial decision tree), where μ is the smallest integer satisfying the relation BIN(μ+1,**b**)≥**N**. Since

$$\Sigma_{j=0,\mathbf{b}}\,\mathrm{BIN}(t,j) \geq \mathrm{BIN}(t,\mathbf{b})+\mathrm{BIN}(t,\mathbf{b}-1) =\mathrm{BIN}(t+1,\mathbf{b}),$$

it follows that t<μ whenever **b**>1; when **b**=1, the two solutions coincide.

The <*,*,**b**>-game is exactly the *unbounded-searching problem with variable-costs* discussed in [4] where an explicit solution was given only for **b**≤2; implicit in the paper was that, for **b**>2, an upperbound on the unknown value **x** should first be established using g(j)=BIN(j,**b**) (see sect. 3.3) and then the problem solved using their technique for the bounded case with the ramaining **b**-1 overestimates. This would yield a solution to the <*,*,**b**>-game requiring 2μ-1 questions, where μ is the smallest integer satisfying the relation BIN(μ+1,**b**)≥**x**. The solution presented in this paper requires 2t-1 questions, where t is the smallest integer such that f(t,**b**)≥**x**. As for the previous game, the two solutions coincide for **b**=1; in all other cases t < μ.

5.2 Symmetry Breaking

As mentioned in the introduction, the election problem cannot be deterministically solved in an anonymous synchronous network [2]; probabilistic solutions to this problem (also known as *symmetry breaking*) do however exist [10,13,22]. In particular, solutions have been proposed for a special class of networks which includes *rings* and *complete graphs* [10,22]; these algorithms terminate with probability 1 and assume that both the number **n** of processors and the girth **g** of G is known to the processors; except for the case of *rings*, simultaneous initiation is also assumed. The employed strategy is composed of a sequence of random selection rounds: at the beginning of each round, every processor randomly selects an integer in [1,**n**]; the round consists now of finding the minimum of the chosen values and determinining if this value is unique; if so, the processor that has chosen this value becomes *elected* and the algorithm terminates; otherwise, a new round of random selection is started. The expected number of rounds of this strategy is e-1=1.718. If a simple 'solitude' verification mechanism is added to the minimum-finding algorithm proposed here (to detect if the smallest value is unique), this algorithm can be employied in each round of random selection yielding a new symmetry-breaking algorithm for this class of graphs; this new symmetry breaking algorithm would terminate with probability 1 using O(k **e**) bits in time O(k $\mathbf{n}^{1/k}$ **g**) on the average for any positive integer k. For a fixed k, this bound matches the best bound established for this problem [10].

5.3 Other Applications

In [4], Benteley and Brown describe several problems where the use of binomial trees would yield an improvement over existing solutions. A further improvement can be obtained by using Binomial-Sum trees instead of binomial trees in at least the following problems: *reduced sensitivity analysis*, *broadcasting to points on a line*, and *linear recursion*. The definitions and details of these problems can be found in [4]; the understanding of the improvement obtained using BS trees in these problems is left as an exercise to the reader.

6. CONCLUDING REMARKS AND OPEN PROBLEMS

In this paper, an improved solution algorithm has been presented for minimum-finding in anonymous synchronous networks, and used to derive improved bounds for the election and spanning-tree construction problems in synchronous networks with distinct values. It has been shown that the proposed minimum-finding algorithm exhibits a time *vs* communication trade-off which is optimal, at any point of the trade-off, among a class of solution algorithms. These results have been derived by reducing minimum-finding to a number-guessing game, and establishing upper and lower bounds on the complexity of the solution strategy. The guessing game considered here is strictly related to the searching problems with variable-cost comparisons investigated by Bentley and Brown [4]; the proposed solution strategy yields an improved solution for those searching problems and their applications.

The proposed solution for minimum-finding in anonymous synchronous netwoks has been developed under two assumptions: *knowledge of* ***n*** and *simultaneous initiation.*

The attentive reader might have already observed that, in sections 2 and 4, knowledge of **n** can be replaced by knowledge of the diameter $\boldsymbol{\partial}$(G) of the network; in such a case, $\boldsymbol{\partial}$(G) can replace **n** in the time bounds stated by theorems 5 and 6. That some knowledge of $\boldsymbol{\partial}$(G) is needed seems to be implied in the proof (by Attiya, Snir and Warmuth [3]) that, in an anonymous *ring* network, non-costant functions (e.g., *min*) cannot be computed without any knowledge of the ring size. However, *exact* knowledge of either $\boldsymbol{\partial}$(G) or **n** is not actually required: any value $\mathbf{m} \geq \boldsymbol{\partial}$(G) would do in the proposed algorithm, provided that this value is available to all processors. An interesting open question is the following: regardless of the complexity, is knowledge of some upperbound on $\boldsymbol{\partial}$(G) really *necessary* to solve this problem?

As for simultaneous initiation, this condition is crucial for the correct functioning of the proposed solution. If this condition does not hold, it is always possible to bound the delay between initiation times to at most $\boldsymbol{\partial}$(G); it is however an open problem to find the minimum value under these conditions in less than O(**i n**) time with a linear number of bits.

REFERENCES

[1] K. Abrahamson, A. Adler, R. Gelbart, L. Higham, D. Kirkpatrick, "The bit complexity of probabilistic leader election on a unidirectional ring", *Proc. 1st Int. Workshop on Distributed*

Algorithms on Graphs, Aug. 1985, to appear.

[2] D. Angluin, "Local and global properties in networks of processes", *Proc. 12th ACM Symp. on Theory of Computing*, April 1980, 82-93.

[3] C. Attiya, M. Snir, M. Warmuth, "Computing on an anonymous ring", *Proc. 4th ACM Symp. on Principles of Distributed Computing*, Aug. 1985, 196-204.

[4] J.L. Bentley, D.J. Brown, "A general class of resource tradeoffs", *Proc. 21th Symp. on Foundations of Computer Science*, Oct. 1980, 217-228.

[5] H.L. Brolaender, J. van Leeuwen, "New upperbounds for distributed extrema-finding in a ring of processors", *Proc. Int. Workshop on Distributed Algorithms on Graphs*, Aug. 1985.

[6] J. Burns, "A formal model for message passing systems", TR-91, Indiana University, Sept. 1981.

[7] E.J. Chang, "Echo algorithms: depth parallel operations on general graphs", *IEEE Trans. Softw. Eng. SE-8*, 1982, 391-400.

[8] D. Dolev, M. Klawe, M. Rodeh, "An O(n log n) unidirectional algorithm for extrema-finding in a circle", *J. Algorithms 3*, 1983, 245-260.

[9] G.N. Frederickson, N. Lynch, "The impact of synchronous communication on the problem of electing a leader in a ring", *Proc. 16th ACM Symp. Theory of Computing*, April 1984, 493-503.

[10] G.N. Frederickson, N. Santoro, "Symmetry breaking in synchronous networks", *Proc. 2nd Int. Workshop on Parallel Computing and VLSI*, July 1986, to appear.

[11] E. Gafni, "Improvements in the time complexity of two message-optimal election algorithms", *Proc. 4th ACM Symp. on Principles of Distributed Computing*, Aug. 1985, 175-185.

[12] R.G. Gallager, "Finding a leader in a network with O(e)+O(n log n) messages", MIT Internal Memo, 1979.

[13] A. Itai, M. Rodeh, "Symmetry breaking in distributive networks", *Proc. 22nd IEEE Symp. on Foundations of Computer Science*, Oct. 1981, 150-158.

[14] E. Korach, S. Moran, S. Zaks, "Tight lower and upper bounds for some distributed algorithms for a complete network of processors", *Proc. 3rd ACM Symp. Princ. Distr. Comput.*, 1984, 199-207.

[15] E. Korach, D. Rotem, N. Santoro, "Distributed election in a circle without a global sense of orientation", *Int. J. Comput. Math. 16*, 1984, 115-124.

[16] M.C. Loui, T.A. Matsushita, D.B. West, "Election in a complete network with a sense of direction", *Information Processing Letters*, 1986.

[17] A. Marchetti-Spaccamela, "New protocols for the election of a leader in a ring", *Proc. 5th Conf. on Foundations of Software Technology and Theoretical Computer Science*, Dec. 1985, to appear.

[18] M. Overmars, N. Santoro, "Tradeoffs for distributed elections in synchronous rings", CS-TR-97, Carleton University, June 1986.

[19] J. Pachl, D. Rotem, E. Korach, "Lower bounds for distributed maximum finding algorithms", *J. ACM 31*, 1984, 380-401.

[20] G.L. Peterson, "An O(n log n) unidirectional algorithm for the circular extrema problem", *ACM Trans. Prog. Lang. Syst. 4*, 1982, 758-762

[21] N. Santoro, "On the message complexity of distributed problems", *Int. J. Comput. Inf. Sci. 13*, 1984, 131-147.

[22] N. Santoro, D. Rotem, "On the complexity of distributed elections in synchronous graphs",

Proc. 11th Int. Workshop on Graphtheoretic Concepts in Computer Science, June 1985, 337-346.

[23] P. Vitanyi, "Distributed elections in an Archimedean ring of processors", *Proc. 16th ACM Symp. Theory of Computing*, April 1984, 542-547.