Determinism vs. Nondeterminism in Multiparty Communication Complexity

Danny Dolev
Computer Science Department
Hebrew University
and
IBM Research
Almaden Research Center
650 Harry Road
San José, California 95120

Tomás Feder*
Computer Science Department
Stanford University
Stanford, California 94305

February 7, 2002

Abstract

A given Boolean function has its input distributed among many parties. The aim is to determine which parties to talk to and what information to exchange in order to evaluate the function while minimizing the total communication. This paper shows that it is possible to evaluate the Boolean function deterministically with only a polynomial increase in communication and number of parties accessed with respect to the information lower bound given by the nondeterministic communication complexity of the function.

1 Introduction

Our model of multiparty communication complexity is motivated by two basic earlier models. The two-party communication model assumes that each of two processors has a part of the input, and the aim is to compute a function on the input minimizing the amount of communication. In the decision tree model, the input is distributed among many memory locations, and the aim is to compute a function on the input while minimizing the number of memory locations examined. Our multiparty communication model extends these two basic models by assuming that the input is distributed among many processors; here the goal is to minimize both communication and number of processors accessed.

Two-party communication has been extensively studied. The main issues studied were the relative power of determinism, nondeterminism and randomization. Yao[19] introduced the tool of minimum fooling set (or crossing sequence) as a measure for the amount of information that needs to be exchanged for a given input partitioned among the two parties. The same technique was widely used in [2, 7, 9, 11, 13].

The decision tree model has been studied in several contexts [3, 10, 12, 15, 16, 17]. An area that inspired research in this direction is the study of graph properties (see [14] for example). The

*Supported by an AT&T Bell Laboratories Scholarship.

main focus in these studies is how to minimize the fraction of the input that must be examined in order to verify a given property. Here again we are interested in the relative power of determinism, nondeterminism and randomization. The basic issue is how to decide what input locations to examine. Similar reduction ideas appear in the proof of Theorem 1 in [1].

In the multiparty communication model, when a large amount of information is distributed among a large number of processors, it is crucial to decide both which processors to communicate with and what information to exchange. We can neither talk to all parties as in the two-party model, nor obtain all the information known to each party as in the decision tree model. A natural measure for the least amount of information required is the information that a nondeterministic algorithm needs to exchange in order to decide the value of the function. In this paper we show that when computing a Boolean function, this information can be obtained deterministically with limited overhead. More precisely, we prove that the deterministic and the nondeterministic communication complexity of multiparty boolean function evaluation are polynomially related.

Tight bounds relate the deterministic and the nondeterministic communication complexity in the two-party model. Let C_1 be the nondeterministic communication complexity of the language defined by a boolean function $f(x_1, x_2)$, and C_0 that of its complement. Aho, Ullman and Yannakakis [2] showed that the deterministic complexity of f is at most $O(C_0C_1)$; Halstenberg and Reischuk [9] improved this bound to $C_0C_1(1+o(1))$. A matching lower bound was obtained by Halstenberg and Reischuk [9], improving an earlier result of Melhorn and Schmidt [11]. Fürer [8] obtained similar lower bounds for the randomized case. Further restrictions on the communication exchange such as bounding the number of rounds have been studied by Papadimitriou and Sipser [13], Duris, Galil and Schnitger [7] and others.

Quadratic bounds relating deterministic and nondeterministic complexities have also been obtained for decision trees. Let k_1 and k_0 be the nondeterministic complexity (the number of memory locations examined) of a boolean function of n variables $f(x_1, \ldots, x_n)$ and its complement. Blum and Impagliazzo [3] Hartmanis and Hemachandra [10], and Tardos [17], independently showed that the deterministic decision tree complexity of f is at most k_0k_1. Related results for randomized decision trees can be found in Saks and Wigderson [16] and Nisan [12].

Our work was motivated by the striking similarity of the results in these two models, which give quadratic C_0C_1 and k_0k_1 bounds, respectively. The methods used to obtain the bounds in these two models, however, are very different. Since in distributed computing, the natural model is one that combines both, one should wonder whether a similar relation holds for multiparty communication. Our result gives a bound on the order of ${k_0}^2k_1$ for the number of parties accessed with C_0C_1 bits exchanged with each one, up to logarithmic factors, where k_1 and C_1 are the number of parties accessed and the total number of bits exchanged in a nondeterministic algorithm for f, and k_0 and C_0 are the analogous parameters for the complementary function $1-f$. This bound essentially matches the communication bound for the two-party case while only increasing the bound on the number of parties accessed by a factor of k_0 with respect to the decision tree case. It improves the bound $(k_0C_0)^2(k_1C_1)$ on the total communication from an earlier version of this paper by a factor of C_0 [6].

Communication complexity in distributed computing has mainly focused on the number of messages or bits required to compute a specific function in a system. The complexity usually arises from either symmetry breaking or asynchronous behavior. The only study that is somewhat close to ours was done by Tiwari [18]. Tiwari studies mainly a chain of processors computing a function

$f(x_1, x_2)$, where the inputs are at both ends of the chain. The difficulties in this model are knowing what information to distribute (as in the two-party model) and how that information should be propagated along the chain. In this model the added complexity of deciding what processors to query does not arise.

In order to concentrate on the combined complexity of deciding what processors to query and what information to exchange with them, we assume the following model. The input is distributed among n parties, and a single *coordinator* can communicate directly with each one of them. One can easily show that allowing direct communication among the parties will not significantly affect the bounds that we obtain.

In [5] a different communication complexity model was defined. In that model each party has all the inputs but one, and all parties communicate through a shared "blackboard." This model was also used in [4]. Our results do not apply to this model because the inputs that individual parties hold are not independent.

2 Definitions

Suppose that a coordinator wishes to evaluate a boolean-valued function $f(x_1, \ldots, x_n)$, where each x_i is chosen from an arbitrary set Γ_i. The input vector $x = (x_1, \ldots, x_n)$ is distributed among n parties, with x_i known to party i.

We shall define nondeterministic algorithms in terms of communucation behavior. A nondeterministic algorithm $\mathcal{N}_1$ that accepts the language defined by f (the set of input vectors that map to 1 under f) is a tuple $(S_1, \ldots, S_n, A_1, \ldots, A_n, V^1)$. The components of such a tuple are as follows. Each S_i is a set of nonempty binary sequences that represents the possible communication exchanges between the coordinator and party i. The binary sequences in S_i are *self-delimiting*, i.e., no one is a prefix of another (this makes it possible to uniquely determine the end of the sequence). Each A_i is a function that maps each sequence $s_i \in S_i \cup \{\epsilon\}$ to a nonempty subset $A_i(s_i)$ of Γ_i; this subset represents the possible inputs at party i for which s_i is a valid communication from the point of view of party i. Here ϵ is the empty sequence and represents the case where no communication occurs between the coordinator and party i; we thus require that $A_i(\epsilon) = \Gamma_i$. A *communication vector* $s = (s_1, \ldots, s_n)$ with $s_i \in S_i \cup \{\epsilon\}$ *covers* an input vector $x = (x_1, \ldots, x_n)$ at party i if $x_i \in A_i(s_i)$. Furthermore, x is *consistent* with s if s covers x at each party i. We say that party i is *accessed* by s if s_i is non-empty. The communication vector s is a 1-*certificate* if $f(x) = 1$ for all x consistent with s. The last component V^1 is a set of 1-certificates such that each input vector x with $f(x) = 1$ is consistent with some $s \in V^1$, and represents the communication vectors that are accepted by the coordinator.

We characterize the communication complexity of $\mathcal{N}_1$ with two parameters. The first parameter C_1 is the maximum over all 1-certificates $s \in V^1$ of $\sum_i \mathrm{length}(s_i)$; thus C_1 is the maximum number of bits exchanged when $\mathcal{N}_1$ accepts. The second parameter k_1 is the maximum over all 1-certificates $s \in V^1$ of the number of parties accessed by s; thus k_1 is the maximum number of parties accessed when $\mathcal{N}_1$ accepts. We also assume the existence of a nondeterministic algorithm $\mathcal{N}_0$ that accepts the language defined by the complementary function $1 - f$, and define 0-certificates, V^0, C_0, k_0 and the appropriate terminology analogously.

We say that a 1-certificate s and a 0-certificate t are *incompatible* at party i if $A_i(s_i) \cap A_i(t_i) = \emptyset$. Notice that every 0-certificate must be incompatible with every 1-certificate somewhere, because otherwise we could construct an input vector on which f takes both values 0 and 1.

3 A Deterministic Algorithm

The algorithm of Blum and Impagliazzo [3] for the decision tree model works by repeatedly 'exposing' the parties accessed by given 1-certificates in turn; each 1-certificate chosen for this purpose is required to cover the input at parties exposed earlier by previous 1-certificates. By incompatibility, if t is a 0-certificate that covers the input at the parties already exposed, then the next 1-certificate s chosen must expose a new party accessed by both s and t. Thus by the time k_0 1-certificates have been chosen, any 0-certificate consistent with the input has been completely exposed, and the value of f can be verified directly. The total number of parties exposed is at most $k_0 k_1$.

A straightforward adaptation of this approach does not work in our model. The reason is that it is too expensive to obtain all the information stored at each party exposed. To overcome this difficulty, we choose a set of parties to expose. Each party exposed evaluates with respect to its input those 1-certificates that were not yet discarded. It communicates enough information, via a 0-certificate that covers its input, to discard a fraction of the possible 1-certificates left. To keep the amount of information "wasted" bounded, it does not communicate when this implies discarding only a very small fraction. Only when no exposed party has a valuable contribution does the coordinator use the remaining 1-certificates to choose more parties to expose. Every time the set of exposed parties increases, the number of exposed accessed parties for each 0-certificate consistent with the input increases as well, as in the decision tree algorithm. By the $k_0 + 1$-th time the value of the function is determined.

The following lemma will be important in bounding the amount of communication required by the algorithm.

Lemma 3.1 *If the Boolean function f has nondeterministic complexity bounded by C_0, k_0, C_1, k_1, then there exists a nondeterministic algorithm for f for which the set of 1-certificates satisfies $|V^1| \leq 2^{C_1} {k_0}^{k_1}$.*

Proof. The number of 1-certificates in V^1 is at most $2^{C_1} n^{k_1}$, since each certificate s is described by the C_1 bits communicated and the k_1 out of n parties accessed. We show that the dependency on n can be eliminated by replacing n with the potentially smaller k_0. Consider the list of all 0-certificates in V^0 in some canonical order (say, the lexicographic order). Choose a 1-certificate s in V^1, and produce the following description. For each 0-certificate t in the canonical list in turn, find a party i at which s is incompatible with t, indicate which of the k_0 parties accessed in t is party i, and then give the sequence s_i that characterizes the communication with party i. Delete then from the list all 0-certificates that are incompatible with s at party i. When the end of the list is reached, the description contains at most C_1 communication bits and k_1 parties described by a number in the range $1, \ldots, k_0$, for a total of $2^{C_1} {k_0}^{k_1}$ possible descriptions. The communication vector s' indicated by this description may be smaller than the 1-certificate s, since only a fraction of the parties accessed by s is listed in the description. On the other hand, by construction, each 0-certificate t in V^0 is incompatible with s', so s' is indeed a 1-certificate. The certificate s' can in fact be recovered from the description by traversing the canonical list and identifying the appropriate parties. Thus the number of 1-certificates s' obtained by this construction is indeed bounded by $2^{C_1} {k_0}^{k_1}$. □

We now describe a deterministic algorithm for a Boolean function f. In this algorithm, all communication is initiated by the coordinator, who sends messages to various parties in turn and receives a response from each of them. Just like in the conventional two-party model, each party knows the protocol in advance and uses its own local memory during the execution. When the algorithm terminates, the coordinator must hold the value of f.

Theorem 3.1 *There is a deterministic algorithm for f that communicates with a total of $2k_0^2k_1$ parties, and exchanges $2(C_1 + \lceil k_1 \log k_0 \rceil + 1)(C_0 + k_0(\lceil \log(2k_0^2k_1) \rceil + 2))$ bits with each.*

Proof. The deterministic algorithm for computing $f(x_1, \ldots, x_n)$ maintains two sets: a set of chosen parties, the *exposed* parties, and a set of candidate 1-certificates from V^1, the *current* 1-certificates. The algorithm runs in $k_0 + 1$ phases, and satisfies the following basic properties:

- i. All communication during a phase occurs only between the coordinator and exposed parties.
- ii. All information sent by an exposed party to the coordinator is shared with all of the exposed parties, so that every exposed party can deduce the set of current 1-certificates.
- iii. New parties are exposed only at the end of a phase.
- iv. If the value of the function is 0 then at the beginning of phase j, each 0-certificate consistent with the input accesses at least j exposed parties.

Each phase discards some 1-certificates that are not consistent with the input vector $x = (x_1, \ldots, x_n)$, by communicating 0-certificates that cover the input at some exposed party to all other exposed parties. If it is no longer possible to discard a large fraction of the 1-certificates in this way with a reduced amount of communication, then we shall show that the following property must hold: each 0-certificate t in V^0 consistent with the input must be incompatible with at least half of the current 1-certificates at *non-exposed* parties. This property implies that such a t must be incompatible with at least a fraction $1/(2k_0)$ of the current 1-certificates at *some* non-exposed party (since at most k_0 parties are accessed by t). We then expose all non-exposed parties accessed by a fraction $1/(2k_0)$ of the current 1-certificates; this exposes, in particular, at least one more party accessed by t, for each 0-certificate t in V^0 consistent with the input. If the set of current 1-certificates is still non-empty, we proceed to the next phase.

By the time the $(k_0 + 1)$-th phase is executed, if all 1-certificates have been discarded, then the value of f is 0, otherwise $k_0 + 1$ parties are accessed by every 0-certificate t in V^0 consistent with the input; this is impossible unless no such certificate exists, in which case the value of f is 1.

Each phase thus consists of two steps: The first step reduces the number of current 1-certificates, the second step increases the number of exposed parties (and implicitly the number of exposed parties for 0-certificates consistent with the input). The two steps are given below in full detail. Note that at the beginning of the first phase, step 1 can be skipped since no exposed parties have been chosen yet, and that step 2 need not be executed in the (k_0+1)-th and last phase, because by that time the value of f is already determined by whether the set of current 1-certificates is empty or not.

1. Each exposed party i in turn looks for a 0-certificate t in V^0 such that t covers the input at party i and t is incompatible at party i with at least $1/(2\alpha)$ of the current 1-certificates per bit needed to describe t at party i, for α as specified below. We shall see that the number of bits needed is $\text{length}(t_i) + \lceil \log(2k_0^2k_1) \rceil + 2$, so t must be incompatible with at least $(\text{length}(t_i) + \lceil \log(2k_0^2k_1) \rceil + 2)/(2\alpha)$ of the current 1-certificates at party i. Party i communicates such a certificate, if found, to the coordinator, in which case each exposed

party is told this t_i and updates the set of current 1-certificates accordingly (the 1-certificates incompatible with t at party i are discarded). The next exposed party is now considered, in a round-robin fashion.

2. If no 1-certificates can be discarded as just described, then each 0-certificate that contains the input will be incompatible with at least half of the current 1-certificates at non-exposed parties. The coordinator and each exposed party can recognize this situation, find all the parties accessed by a fraction of at least $1/(2k_0)$ of the current 1-certificates, and add these parties to the set of exposed parties. Now each 0-certificate that contains the input has one more accessed party exposed.

The communication bound is obtained as follows. Since each bit communicated with a given exposed party discards at least $1/(2\alpha)$ of the current 1-certificates, 2α bits must discard more than half of the current 1-certificates. By the bound in the lemma, this halving can be done at most $C_1 + \lceil k_1 \log k_0 \rceil$ times before all 1-certificates have been discarded. Adding another 2α bits to ensure that the description of the last 0-certificate used to discard 1-certificates is not truncated, we obtain a $(C_1 + \lceil k_1 \log k_0 \rceil + 1)(2\alpha)$ bound on the communication with each exposed party. With α as defined below, one can check that α is indeed at least as large as the description of a certificate at a party, and that the communication bound in the statement of the theorem is satisfied.

We shall see below that at most $2k_0k_1$ parties are exposed at each phase, for a total of $2k_0^2k_1$ parties over the entire execution of the algorithm (since we need not expose parties at phase k_0+1). If this bound is maintained, then a 0-certificate t in V^0 at party i can be described with length(t_i) bits, plus an additional $\log(2k_0^2k_1)$ bits to identify i within the set of current exposed parties. In communicating this information to each exposed party j, two additional bits are used: one bit is used by party j to tell the coordinator whether, after 1-certificates have been discarded according to t, there is some new 0-certificate t' which party j can use to discard 1-certificates; and one bit is sent back by the coordinator to tell party j whether it wants to use this new t' as the next 0-certificate to discard 1-certificates. Thus the communication of t_i to each party costs length$(t_i) + \lceil \log(2k_0^2k_1) \rceil + 2$ bits.

We choose $\alpha = (C_0 + k_0(\lceil \log(2k_0^2k_1) \rceil + 2))$. If no exposed party i can provide a 0-certificate t in V^0 that covers the input at party i and is incompatible with a fraction of at least $\rho_i = (\text{length}(t_i) + \lceil \log(2k_0^2k_1) \rceil + 2)/(2\alpha)$ of the current 1-certificates at party i, then every 0-certificate t in V^0 consistent with the input is incompatible with at most $\Sigma_i \rho_i \le 1/2$ of the current 1-certificates at exposed parties, where the sum is over the parties accessed in t (at most k_0 of them). Hence every 0-certificate t in V^0 consistent with the input must be incompatible with at least half of the current 1-certificates at non-exposed parties, as claimed.

Since at most k_1 parties are accessed by a single 1-certificate, the sum over all parties of the fraction of current 1-certificates that access them is k_1, and so the number of parties accessed by a fraction of at least $1/(2k_0)$ of these current certificates is at most $2k_0k_1$. This proves the bound on the number of exposed parties added at each phase, completing the proof. □

4 Conclusion and Open Problems

In this paper we studied communication complexity in a multiparty model. The approach is based on the two-party model and the decision tree model. Some results from the two basic models can be applied to our model. The main open problems are the existence of lower bounds in this model and the study of randomization. An intriguing question is whether a quadratic upper bound with $O(k_0k_1)$ parties accessed and with polynomial communication can be achieved. The study of other measures, such as the number of phases [7, 13], and of more general communication networks [18], has a special importance for understanding communication in distributed systems.

Acknowledgements

The authors are very grateful to Rüdiger Reischuk for his careful reading of the paper and his helpful comments.

References

[1] L. Adelman, "Two Theorems on Random Polynomial Time", Proc 19th FOCS, 75–83, 1978.

[2] A.V. Aho, J.D. Ullman and M. Yannakakis, "On Notions of Information Transfer in VLSI Circuits," Proc. 15th ACM STOC, 133-139, 1983.

[3] M. Blum and R. Impagliazzo, "Generic Oracles and Oracle Classes," Proc. 19th ACM STOC, 118-126, 1987.

[4] L Babai, N. Nisan, and M. Szegedy, "Multiparty Protocols and Logspace-hard Pseudorandom Sequences," Proc. 21th ACM STOC, 1-11, 1989.

[5] A. Chandra, Furst, and R. Lipton, "Multiparty Protocols," Proc. 24th IEEE FOCS, 1983.

[6] D. Dolev and T. Feder, "Multiparty Communication Complexity," Proc. 30th IEEE FOCS, 428-433, 1989.

[7] P. Duris, Z. Galil and G. Schnitger, "Lower Bounds on Communication Complexity," Proc. 16th ACM STOC, 81-91, 1984.

[8] M. Fürer, "The Power of Randomness for Communication Complexity," Proc. 19th ACM STOC, 178-181, 1987.

[9] B. Halstenberg and R. Reischuk, "On Different Modes of Communication," Proc. 20th ACM STOC, 162-172, 1988.

[10] J. Hartmanis and L. H. Hemachandra, "One-way Functions, robustness, and non-isomorphism of NP-complete sets," Technical report DCS TR86-796, Cornell University, 1987.

[11] K. Melhorn and E. M. Schmidt, "Las Vegas is better than Determinism in VLSI and Distributed Computing," Proc. 14th ACM STOC, 330-337, 1982.

[12] N. Nisan, "CREW PRAMs and Decision Trees," Proc. 21st ACM STOC, 327-335, 1989.

[13] C.H. Papadimitriou and M. Sipser, "Communication Complexity," Proc. 14th ACM STOC, 196-200, 1982.

[14] R. Rivest and S. Vuillemin, "On Recognizing Graph Properties from Adjacency Matrices," Theor. Comp. Sci. 3, 371-384, 1978.

[15] M. Snir, "Lower Bounds for Probabilistic Linear Decision Trees," Theor. Comp. Sci. 38, 69-82, 1985.

[16] M. Saks and A. Wigderson, "Probabilistic Boolean Decision Trees and the Complexity of Evaluating Game Trees," Proc. 27th IEEE FOCS, 1986.

[17] G. Tardos, "Query complexity, or why is it difficult to seperate $\mathcal{NP}^A \cap \mathrm{co}\mathcal{NP}^A$ from $\mathcal{P}^A$ by a random oracle A?" Manuscript, 1988.

[18] P. Tiwari, "Lower Bounds on Communication Complexity in Distributed Computer Networks," JACM, vol. 34 no. 4, 921-938, 1987.

[19] A. Yao, "Some Complexity Questions Related to Distributive Computing," Proc. 11th ACM STOC, 209-213, 1979.