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

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 with each of them in order to evaluate the function while minimizing the total communication. This paper shows that it is possible to obtain the Boolean answer deterministically with only a polynomial increase in communication with respect to the information lower bound given by the nondeterministic communication complexity of the function.

1 Introduction

The question of multi-party 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. The multi-party 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 in the last decade. The basic emphasis was inspired by VLSI complexity, and this lead to the study of two-party communication. The main issues studied were the relative power of determinism, nondeterminism and randomization. Yao[14] 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, 4, 6, 7, 9]. The essence of this technique amounts to identifying what information to exchange in order to minimize the amount of communication.

The decision tree model has been studied in several contexts [3, 8, 11, 12]. An area that inspired research in this direction is the study of graph properties (see [10] for example). The 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 multi-party 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 with each one. We can neither talk to all parties as in the two-party model, nor obtain all

*Supported by an AT&T Bell Laboratories Scholarship.

the information known to each party when we talk to it 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 multi-party 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 [6] improved this bound to $C_0C_1(1 + o(1))$. A matching lower bound was obtained by Halstenberg and Reischuk [6], improving an earlier result of Melhorn and Schmidt [7]. Fürer [5] 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 [9], Duris, Galil and Schnitger [4] 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] 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 [12] and Nisan [8].

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 multi-party communication. Our result gives a bound on the number of bits exchanged by a deterministic algorithm for a boolean function f of the order of $(k_0C_0)(k_1C_1)\ 2$, up to logarithmic factors, where k_1 and C_1 are the number of processors accessed and the number of bits exchanged in a nondeterministic algorithm for f, and k_0 and C_0 are the analogous parameters for the complementary function $1 - f$.

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 [13]. 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 obtaining his result are knowing what information to distribute (as in the two-party model) and how that information should be propagated along the chain. However, in his 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.

2 Definitions

Suppose a coordinator wishes to evaluate a boolean function $f(x_1, \ldots, x_n)$. The input vector $x = (x_1, \ldots, x_n)$ is distributed among

n parties, with x_i known to party i, where x_i is chosen from a set Γ_i.

We shall assume the existence of a nondeterministic algorithm $\mathcal{N}_1$ that accepts the language defined by f (when the value of f is 1). The communication behavior of the nondeterministic algorithm can be described by a *communication vector* $s = (s_1, \ldots, s_n)$, where s_i is a self-delimiting binary communication sequence between the coordinator and party i, for every i. Each possible run of the nondeterministic algorithm $\mathcal{N}_1$ has a corresponding communication vector $s = (s_1, \ldots, s_n)$. Let $A_i(s_i)$ be the set of all x_i for which s_i is a valid communication sequence under $\mathcal{N}_1$ between the coordinator and party i, when this party holds input x_i, from the point of view of party i. In particular, if s_i is the empty sequence, then $A_i(s_i) = \Gamma_i$; if s_i is non-empty, then we say that party i is *accessed* in s. A communication sequence $s = (s_1, \ldots, s_n)$ is a 1-*certificate* if the algorithm *accepts* the input when the communication with the n parties is given by s, under the protocol $\mathcal{N}_1$. We say that a 1-certificate $s = (s_1, \ldots, s_n)$ *covers* an input vector $x = (x_1, \ldots, x_n)$ at party i if $x_i \in A_i(s_i)$. Furthermore, s *contains* x if s covers x at each party i.

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 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 of the number of parties accessed in 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, 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 parties accessed under given 0-certificates; each 0-certificate chosen covers the input at parties exposed earlier. By incompatibility, for every 1-certificate s that covers the input at the parties already exposed, each 0-certificate chosen exposes a new party with a non-empty communication under s. Thus by the time k_1 0-certificates have been chosen, every 1-certificate that could contain the input has been exposed, and the value of f can be easily verified. 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 (in its turn) evaluates those 1-certificates that were not discarded yet with respect to its input. 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 it implies discarding only a very small fraction. Only when this is no longer possible, so that no exposed party has a valuable contribution, do we choose a new 0-certificate that exposes more parties. By the time k_1 0-certificates have been chosen, the number of 1-certificates left has been halved. This is an adaptation of the two-party deterministic algorithms.

Theorem 3.1 *There is a deterministic algorithm for f that runs in $C_1 + k_1 \lceil \log k_0 \rceil$ phases, talks to $k_0 k_1$ parties in each phase,* *and communicates* $(C_1 + k_1\lceil\log k_0\rceil)(k_1)(C_0 + k_0\lceil\log(k_0k_1)\rceil)$ *bits per party per phase.*

Proof. We assume that the set of 0-certificates and 1-certificates is minimal, i.e., no certificate contains a smaller certificate (one accessing fewer parties). We shall show that the number of 1-certificates is at most $2\ C_1 + k_1\lceil\log k_0\rceil$. To achieve this, we find a description for every 1-certificate s whose length is at most $C_1 + k_1\lceil\log k_0\rceil$. Note that C_1 bits are sufficient to describe the bits sent under s; then $\lceil\log n\rceil$ bits could be used to identify each of the k_1 parties accessed, for a total of $C_1 + k_1\lceil\log n\rceil$ bits. We reduce the number of bits used to identify each of the k_1 parties to $\lceil\log k_0\rceil$. Consider the list of all 0-certificates in some canonical order (say, the lexicographic order). For each certificate t in this list in turn, find a party i at which s is incompatible with t (assuming s and t are not already incompatible at some party j found earlier), use $\lceil\log k_0\rceil$ bits to describe which of the k_0 parties accessed under t is party i, and then give the bits that party i communicates under s. When the end of the list is reached, we have a description with at most $C_1+k_1\lceil\log k_0\rceil$ bits for s. It is not hard to show that the 1-certificate can in fact be reconstructed from this description by traversing the list of 0-certificates.

Our deterministic algorithm for computing $f(x_1,\ldots,x_n)$ works in $C_1 + k_1\lceil\log k_0\rceil$ phases. Each phase discards 1-certificates that do not contain the input vector $x = (x_1,\ldots,x_n)$; the number of 1-certificates left, the *current* 1-certificates, is halved at each phase. By the end of the last phase, either a proof that $f(x_1,\ldots,x_n) = 1$ has been found, or all 1-certificates have been discarded, and therefore $f(x_1,\ldots,x_n) = 0$.

During each phase, the coordinator communicates with at most k_0k_1 parties, and obtains at most $\alpha = (k_1)(C_0 + k_0\lceil\log(k_0k_1)\rceil)$ bits from them; he also send the α bits for the current phase and all earlier phases to each party. Thus the communication is at most α times the number of phases per party per phase, as stated. The parties that the coordinator has communicated with in each phase will be called the *exposed* parties. At the beginning of the phase, no parties have been exposed, and there are some number ρ of current 1-certificates.

The algorithm iteratively tries to execute step 1, and only executes step 2 if this fails. If step 2 also fails, then $f(x_1,\ldots,x_n) = 1$. Otherwise, the phase ends when the number of 1-certificates has been halved. This will happen by the time the number of bits exchanged in step 1 reaches α, and no later than by the $(k_1 + 1)$th execution of step 2. Note that at the beginning of the phase step 1 is skipped since there are no exposed parties.

1. Each exposed party i in turn looks for a 0-certificate t such that t covers the input at party i and t is incompatible at party i with at least $\rho/2\alpha$ current 1-certificates per bit needed to describe t at party i. The number of bits needed is $\mathrm{length(t_i)} + \lceil\log(\mathrm{k_0k_1})\rceil$, so t must be incompatible with at least $(\mathrm{length(t_i)} + \lceil\log(\mathrm{k_0k_1})\rceil)\rho/2\alpha$ current 1-certificates at party i. It communicates only if such a certificate has been found, in which case the set of current 1-certificates is updated (the 1-certificates incompatible with t at party i are discarded) before the next exposed party is considered.
2. Otherwise, the coordinator finds the first 0-certificate t which is incompatible at non-exposed parties with all current 1-certificates with the exception of a set of at most $\rho/2k_1$ 1-certificates, the *surviving* 1-certificates, and adds the parties accessed in t to the set of exposed parties (there are at most k_0 such parties).

If step 1 fails, and there exists a 0-certificate t that contains the input vector, then this certificate must succeed in step 2, because t is incompatible with every 1-certificate somewhere, and the number of 1-certificates that t is incompatible with at the exposed parties is at most those that were not discarded at step 1. That number is the sum over all exposed i of the expression in step 1, and this is at most $\rho/2k_1$. Thus step 2 can only fail if no 0-certificate contains the input vector, in which case the value of the function is 1. Note that by the time α bits of description have been used in step 1, the number of current 1-certificates has been halved.

The number of surviving 1-certificates after the first j executions of step 2 is at most $(j-1)\rho/2k_1$, because the first time step 2 is executed there are no surviving 1-certificates, and each subsequent execution of step 2 adds at most $\rho/2k_1$ surviving 1-certificates. Also, for each non-surviving 1-certificate s, at least j parties accessed by s have been exposed, since each execution of step 2 exposes at least one more party for each of them. It follows that no current 1-certificate can be non-surviving after the (k_1+1)th execution of step 2, since it would then have to access k_1+1 parties. Thus the number of current 1-certificates after the (k_1+1)th execution of step 2 equals the number of surviving 1-certificates, which is at most $\rho/2$. □

4 Conclusion and Open Problems

In this paper we studied communication complexity in the multi-party model. This model is a natural distributed or parallel model. The adaptation of existing results for the two-party protocols and the decision tree protocols to this model is not straightforward. The difficulties we faced in introducing the deterministic algorithm indicates that this task is feasible but requires further research.

There are some results that can be directly lifted from previous research, for example, results about specific instances or non-existence results. The main open problems are obtaining lower bounds in this model and the study of randomization. An intriguing question is whether a quadratic upper bound exists. The study of other measures, such as the number of phases [4, 9], and of more general communication networks [13], 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 IEEE FOCS, 118-126, 1987.

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

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

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

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

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

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

[10] R. Rivest and S. Vuillemin, "On recognizing graph properties from adjacency matrices," Theor. Comp. Sci. 3, 371-384, 1978.

[11] M. Snir, "Lower bounds for probabilistic linear decision trees," Theor. Comp. Sci. 38, 69-82, 1985.

[12] M. Saks and A. Wigderson, "Probabilistic boolean decision trees and the complexity of evaluating game trees," Proc. 27th IEEE FOCS, 1986.

[13] P. Tiwari, "Lower bounds on communication complexity in distributed computer networks," JACM, vol. 34 no. 4, 921-938, 1987.

[14] A. Yao, "Some complexity questions related to distributive computing," Proc. 11th ACM STOC, 209-213, 1979.