COMMUNICATION COMPLEXITY OF SOME PROBLEMS IN DISTRIBUTED COMPUTATION

by

Zhi-Quan Luo
B.S. Peking University, Beijing
(1984)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF

DOCTOR OF SCIENCE
IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
August 1989

Signature of Author _____
Department of Electrical Engineering and Computer Science
August 7, 1989

Certified by _____
John N. Tsitsiklis
Thesis Supervisor

Accepted by _____
Thomas L. Magnanti
Co-Director, Operations Research Center

TO MY PARENTS, MY BROTHER,
MY SISTER AND MY WIFE.

Acknowledgements

I would like to thank my thesis supervisor Professor John N. Tsitsiklis for his guidance during my Ph.D years at MIT. He has profoundly influenced my scientific thinking and judgement. His care, advice, extensive knowledge and patience (especially towards my writing) has made him an excellent thesis supervisor. I would also like to express my deep gratitude to Professors Dimitri P. Bertsekas and James B. Orlin for their many valuable suggestions concerning my research. Their support and encouragement are also greatly appreciated.

My special thanks go to Professors Steve Kleiman, Gian–Carlo Rota and Michael Artin of MIT for several stimulating discussions. I am also grateful to Professor Peter Olver of University of Minnesota for suggesting the reference [GN 76]. I have also greatly benefitted from the close scientific interactions with my friends Chonghwan Lee, Tom Richardson, Paul Tseng and Siye Wu.

The support and comraderie of my friends in the Laboratory for Information and Decision Systems and at the Operations Research Center have made my stay at MIT a very enjoyable one. Last but not least, I would like to take this opportunity to thank my teachers in Peking Univesity (Beijing, China) for laying the foundation of my mathematical thinking which has led in part to this thesis and will undoubtedly help me in my future career.

This research was carried out in the Laboratory for Information and Decision Systems, with support by Army Research Office under grant DAAL03–86–K–0171.

COMMUNICATION COMPLEXITY OF SOME PROBLEMS IN DISTRIBUTED COMPUTATION

by

Zhi-Quan Luo

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF
DOCTOR OF SCIENCE
IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
AUGUST 1989

Abstract

We consider a situation where two processors P_1 and P_2 are to evaluate a collection of functions $f_1, \ldots, f_s$ of two vector variables x, y, under the assumption that processor P_1 (respectively, P_2) has access only to the value of the variable x (respectively, y) and the functional form of $f_1, \ldots, f_s$. Two kinds of communication protocols are considered, namely, those in which messages are real–valued functions (continuous protocols) and those in which messages are binary strings (discrete protocols). For the class of one–way continuous communication protocols, an almost optimal bound on the communication complexity (the amount of information that has to be exchanged between the processors) is derived when the functions $f_1, \ldots, f_s$ are polynomials. We also derive several new lower bounds for the class of two–way continuous communication protocols which improve on earlier bounds by Abelson [A80]. We then apply our results to the problems such as solving a system of linear equations, solving a polynomial equation and obtain almost tight lower bounds. In contrast, we show, in both cases, that the lower bounds obtained by applying Abelson's result are far from optimal. We also consider the problem of distributed convex optimization where each processor P_i is given a convex function f_i $(i = 1, 2)$ and they wish to, by exchanging binary messages, approximately minimize the sum $f_1 + f_2$. We give both lower and upper bounds on the minimum number of bits that have to be exchanged for this problem. Finally, several issues about multi–party communication are addressed.

Contents

List of Figures

Chapter 1

Introduction

Parallel and distributed processing is becoming increasingly widespread. Its importance stems from the fact that if many computers work simultaneously on one problem they should be capable of solving it much faster than a single computer working alone. This has been common wisdom since the sixties but only recently have parallel computers become commercially available. Alliant, ELXSI, Encore, Thinking Machines, among others, market parallel computers. Many once intractable problems for a single computer can now be solved quickly by a parallel computer. For example, the problem of global weather forecasting requires an extremely large amount of computation consisting of many small loosely related subtasks. This has made it ideal for performing the computation in parallel since each subproblem can be solved by a different processor. In contrast, a serial computer has to work on each subproblem sequentially and the computation sometimes takes so long that a calculated weather prediction may be meaningless by the time the computation finishes. Today, the current sequential computer technology seems to be reaching its limit and the chances for a single serial computer to be able to cope with the above mentioned problem is miniscule . Apart from being technologically infeasible, it is also economically prohibitive to greatly increase the speed of a serial computer. As a result, the idea of parallel processing has burgeoned whereby a computational task is divided up into many parts and a different processor (or a processing unit) is assigned to each part. The processors perform computations cooperatively by transmitting to each other the intermediate results.

Ideally, a parallel computer will work much faster than its sequential counter-

part. However, the performance of a parallel computer depends not only on the nature of the problem under consideration, but also on the machine achitecture. In fact, we are still far from making an effective use of parallelism and many research problems remain unsolved. For instance, problems such as: what is the maximum amount of parallelism achievable for a specific class of problems, what kind of parallel machine achitecture, programming language and compiler are most suitable for a given problem, what kind of achitecture, programming language and compiler are suitable and how to efficiently route messages between different processors, are still not well solved.

The study of parallel and distributed computation differs from that of traditional sequential computation in many respects. For example, communication plays an important role in parallel and distributed processing since information is distributed among many processors. In contrast, this issue is nonexistent for a serial computer where information retrieval and storage is very efficient due to the technology of random access memory. Roughly speaking, the parallel or distributed processing time consists of two parts. The first part is the actual processing time, and the second part is the communication time spent while the processors are waiting for information. In a large distributed computer network, the information exchange can be very costly, especially when the communication resources are scarce. For these reasons we are led to the study of the minimal communication required to solve a given problem.

For example, in the context of distributed decision making where each local agent has only partial information about the environment, communication between agents is crucial for them to jointly make a sensible decision, and the question of efficiently utilizing communication channels naturally arises. A similar scenario occurs in distributed sensing where sensors collect data at spatially distant sites and a decision has to be made based on the total information gathered by all sensors. Since communication can be very time comsuming and expensive, we are led to minimize the information exchange. In certain situations, there may be additional physical limitations on the amount of communications allowed between distinct processors. It is then interesting to ask what is the minimum amount of information that has to be exchanged between each pair of processors, given the

physical constraints. Quite often excessive information exchange among processors can prevent a timely solution to such problems. For the problem of VLSI circuit layout design, the question of minimizing information exchange arises as a result of physical and topological considerations. In particular, the size of a VLSI chip is directly related to the amount of information exchange required for solving a problem under consideration. The less information exchange there is, the smaller the corresponding VLSI chip becomes.

There is yet another reason why the communication issue is so important. With current technology, the local processing can be done very fast and the time spent by each processor doing computation is often negligible compared to the time needed for routing the messages. In other words, most of the processors may be idle most of the time, simply waiting for the desired information to arrive so that the next instruction can be executed. Therefore, to solve a problem efficiently on a parallel computer (or in a distributed computer network) one needs an algorithm that does not require excessive information transfer. The main theme of this thesis, loosely speaking, is to study the minimum amount of information exchange that has to take place in order for a parallel computer (or a distributed computer network) to solve a given problem.

There are many important issues concerning communication in a parallel computer (or in a distributed computer network). For instance, during the course of computation each processor in the network sends and receives messages. Each processor has to determine what information to send, and also when and where to send it. The general principle is that each processor should carry out as much local computation as possible in order to compress the information needed by other processors and send the messages in their most compact form. This is particularly so when the local computation is fast. Of course, the simplest way of exchanging information is to perform a total broadcast among the processors. As a result the problem is converted into a centralized one and the best sequential algorithm can be employed. However, there are several potential drawbacks for performing a total broadcast. First of all, for a given problem, there may be parallel algorithms that are much faster than the best known sequential algorithm. Second, since communication is often very expensive and since in many situations it is the bottleneck to a

timely solution of a problem, performing a total broadcast throughout the computer network may be out of the question. Besides, a total broadcast may be unnecessary because certain data may be irrelavant to some processors. For example, in the context of distributed hypothesis testing, it suffices to communicate only a set of sufficient statistics.

The notion of communication complexity can be roughly defined as the amount of information that has to be communicated in a computer network. Clearly, it should depend heavily on the nature of a given problem, the physical data distribution and the topology of the network. If no processor has complete information, some information exchange is unavoidable during the course of computation. However, issues such as what information to communicate, to whom and when the information should be sent are, nontrivial. In this thesis, we study the inherent communication complexity of certain problems in parallel and distributed computation. The objective of this study is to identify the hard problems which require a lot of information exchange, and to design parallel and distributed algorithms that use minimal communication.

1.1 General Problem Statements

In light of the previous discussion, there are several factors on which communication complexity depends: the nature of the given problem, the physical distribution of the input data among processors, and the topology of the computer network. In most of this thesis, we ignore the topological issues by making the assumption that the computer network consists of only two processors P_1 and P_2. We consider a situation where processors P_1 and P_2 wish to perform some computational task and must communicate because none of them possesses all of the problem data. For the reasons presented before, we will be concerned not with the local processing time of each processor, but instead, with the total information transfer required between the two processors. In measuring the amount of information exchange, two different complexity measures are considered: (1) the continuous framework where the messages are real–valued functions and the amount of communication is measured by the number of real–valued messages exchanged; (2) the discrete

framework in which messages are finite binary strings and we measure the amount of information exchange by the total number of bits used. The continuous model of communication was first introduced and studied by Abelson ([A 78], [A 80]) and was subsequently studied in [LT 89] and [LTs 89]. The discrete model of communication was used to study the communication complexity of computing a Boolean function ([Y 79]) and of approximately computing a continuous function (see [TL 87]). In what follows, we formally describe the continuous/discrete communication protocols and define the corresponding notions of communication complexity.

1.1.1 Communication Protocols, Continuous Model

In this subsection, we introduce a class of continuous communication protocols and formulate the general problem to be studied in Chapters 2 and 3.

Let there be two processors P_1 and P_2. Processor P_1 (respectively, P_2) has access to the value of a vector $x \in \Re^m$ (respectively $y \in \Re^n$). Let there be given a finite collection $\vec{f}$ of functions $f_1, f_2, \ldots, f_s : D_{\vec{f}} \mapsto \Re$, where $D_{\vec{f}}$ is some subset of $\Re^m \times \Re^n$ on which these functions are defined. (For example, if each f_i is a rational function expressed as a ratio of two relatively prime polynomials, it is natural to let $D_{\vec{f}}$ be the set of all vectors at which none of the denominators of these functions vanishes.)

The objective of the processors is to exchange messages and compute the values $f_1(x,y), \ldots, f_s(x,y)$. It is assumed that both processors know the formulas defining these functions. (For instance, if each f_i is a polynomial, then each processor knows the coefficients of these polynomials.) Ideally, a protocol should work for all possible values $(x,y) \in D_{\vec{f}}$ of the "inputs". We will occasionally consider, however, protocols which are defined only when (x,y) belongs to some possibly smaller set $D \subset D_{\vec{f}}$.

In a *two-way communication protocol* π (see Figure 1.1), messages can be exchanged in both directions. We use $r(\pi)$ to denote the number of exchanged messages and we let $T_{1\to 2}$ (respectively, $T_{2\to 1}$) denote the set of i's for which the ith message is transmitted from P_1 to P_2 (respectively, from P_2 to P_1). The protocol is defined in terms of a collection of functions $m_1, \ldots, m_{r(\pi)}$ mapping a set $D \subset D_{\vec{f}}$ into $\Re$. (In particular, $m_i(x,y)$ is the value of the i-th message and the set D is

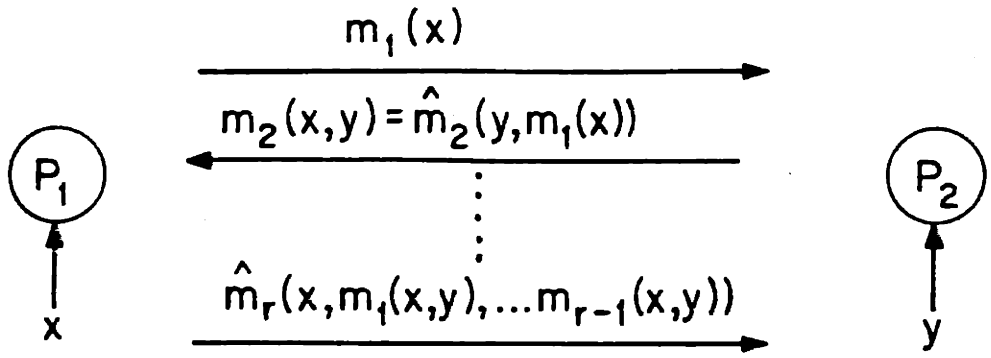

Figure 1.1: Two–way communication protocol.

called the *domain* of the protocol.) Since a message by a processor can only be a function of the information available to that processor, we impose the requirement that for each i, there exists some real–valued function $\hat{m}_i$ such that

$$m_i(x,y) = \hat{m}_i\left(x, m_1(x,y), \ldots, m_{i-1}(x,y)\right), \qquad \forall (x,y) \in D, \quad \text{if } i \in T_{1\to 2}, \quad (1.1.1)$$

and

$$m_i(x,y) = \hat{m}_i\left(y, m_1(x,y), \ldots, m_{k-1}(x,y)\right), \qquad \forall (x,y) \in D \quad \text{if } i \in T_{2\to 1}. \quad (1.1.2)$$

We say that the protocol is *legitimate* if for each i $(i = 1, \ldots, s)$ at least one of the following conditions is true:

a) There exist a function h_i such that

$$f_i(x,y) = h_i\left(x, m_1(x,y), \ldots, m_{r(\pi)}(x,y)\right), \qquad \forall (x,y) \in D. \quad (1.1.3)$$

This corresponds to the case where processor P_1 evaluates f_i.

b) There exists a function h_i such that

$$f_i(x,y) = h_i\left(y, m_1(x,y), \ldots, m_{r(\pi)}(x,y)\right), \qquad \forall (x,y) \in D. \quad (1.1.4)$$

This is the case where f_i is computed by processor P_2.

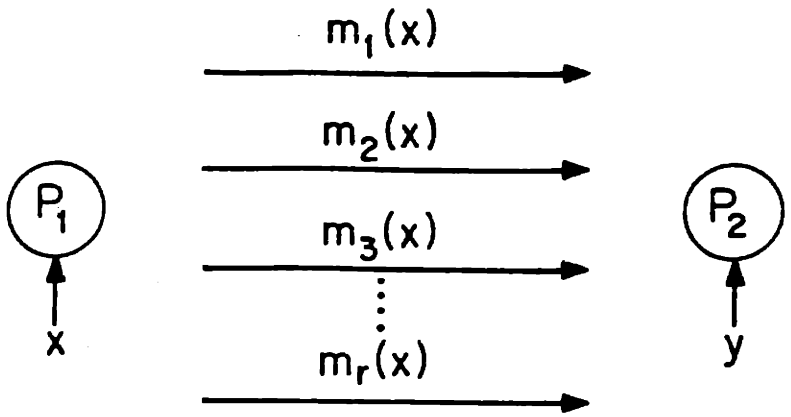

Figure 1.2: One–way communication protocol.

Let $\Pi(\vec{f}; D, \leftrightarrow)$ denote the class of all legitimate two–way protocols, with domain D, for computing the functions $f_1, \ldots, f_s$, subject to some additional restrictions to be introduced later. We define the two–way communication complexity $C(\vec{f}; D, \leftrightarrow)$ for computing $\vec{f}$ on the domain D to be

$$C(\vec{f}; D, \leftrightarrow) = \min_{\pi \in \Pi(\vec{f}; D, \leftrightarrow)} r(\pi). \tag{1.1.5}$$

The definition of an *one–way communication protocol* (see Figure 1.2) π is identical, except that messages can only be transmitted by processor P_1. That is, the set $T_{2\to 1}$ is assumed empty. Let $\Pi(\vec{f}; D, \rightarrow)$ denote the set of all legitimate one–way communication protocols with domain D. We define the one–way (from P_1 to P_2) communication complexity $C(\vec{f}; D, \rightarrow)$ on the domain D to be

$$C(\vec{f}; D, \rightarrow) = \min_{\pi \in \Pi(\vec{f}; D, \rightarrow)} r(\pi). \tag{1.1.6}$$

Notice that in the above models the protocols are "continuous" in the sense that the messages to be sent are real numbers. Given that real numbers can only be encoded with an infinite number of bits, such protocols might seem impossible to implement in practice. However, parallel and distributed numerical algorithms

are almost always described and analyzed as if real numbers can be communicated (e.g. [M 71], [C 76], [M 86]), with the understanding that in practice these numbers will be encoded with a finite number of bits which is sufficient to obtain a desired accuracy. Furthermore, if the messages being transmitted are rational functions of the data and if the data consist of rational numbers, then an implementation using a finite number of bits is clearly possible. Finally, in practice, it is usually the case that a field of a fixed length is used for transmitting an encoded version of a real number. For this reason, it is reasonable to count the number of real–valued messages being transmitted, as opposed to counting individual bits. Our model is therefore a fairly realistic way of capturing the communication resources needed in a number of practical applications.

Typically, some smoothness constraints have to be imposed on the message functions $m_1, \ldots, m_{r(\pi)}$. This is because there exist one–to–one functions from $\Re^m$ into $\Re$, and processor P_1 could transmit the value of its vector x by using a single message. In particular, P_1 could simply interleave the binary expansions of the components of x and use the resulting number as a message. To be more precise, let $x = (x_1, x_2, \ldots, x_m)$ be the input vector to processor P_1. Without loss of generality, we assume

$$0 \leq x_i \leq 1, \qquad \forall i = 1, 2, \ldots, m.$$

Consider the binary expansions of these real numbers

$$\begin{aligned} x_1 &= 0.a_1^1 a_2^1 \cdots a_k^1 \cdots \\ x_2 &= 0.a_1^2 a_2^2 \cdots a_k^2 \cdots \\ &\vdots \\ x_m &= 0.a_1^m a_2^m \cdots a_k^m \cdots . \end{aligned}$$

Let x^* be the real number whose binary expansion is given by

$$x^* = 0.\underbrace{a_1^1 a_1^2 \cdots a_1^m}\,\underbrace{a_2^1 a_2^2 \cdots a_2^m}\,\underbrace{a_3^1 \cdots}\,\cdots .$$

It is not hard to see that x^* is in one–to–one correspondence with $x = (x_1, \ldots, x_m)$. Thus, it suffices to let processor P_1 send the message x^*, since processor P_2, upon receiving this message, can recover the vector x and proceed to compute $f(x, y)$.

This is not a useful protocol, for the purpose of numerical computation, and is unlike any protocol that is used in practice. In contrast to the above described interleaving, a good protocol should compress the information contained in x or y intelligently, and then transmit only the compressed information. For this reason, we shall impose some smoothness requirements on the message functions m_i. From a technical point of view, smoothness assumptions prohibit the use of one–to–one functions from $\Re^m$ into $\Re$, if $m > 1$. From a practical point of view, such smoothness is present in the vast majority of practical numerical methods for algebraic problems. Furthermore, in many parts of this thesis, we concentrate on the case where each one of the functions $f_1, \ldots, f_s$ is rational. It is then natural to restrict attention even further to protocols involving only rational functions of the data. This is equivalent to an assumption that each processor can only perform the elementary arithmetic operations. Such an assumption is common in complexity studies for algebraic problems ([BM 75]).

In the sequel we use the shorter notations $\Pi(\vec{f}; D)$ and $C(\vec{f}; D)$ whenever it is clear from the context whether we are dealing with one–way or two–way protocols. Furthermore, we use the notation $\Pi(f; D)$ and $C(f; D)$ whenever $s = 1$ and the collection $\vec{f}$ of functions consists of the single function f.

In this thesis, we will consider various restrictions on the set of allowed protocols. We indicate these restrictions by using the notation shown in Table 1:

Notations	Restrictions on the message functions $\hat{m}_1, \ldots, \hat{m}_r$ (cf. Eqs. (1.1.1)–(1.1.2)).	Restrictions on the final evaluation functions $h_1, \ldots, h_s$(cf. Eqs. (1.1.3)–(1.1.4)).
$\Pi_0(\vec{f}, D)$	continuously differentiable	continuous
$\Pi_1(\vec{f}, D)$	continuously differentiable	continuously differentiable
$\Pi_2(\vec{f}, D)$	twice continuously differentiable	twice continuously differentiable
$\Pi_\infty(\vec{f}, D)$	infinitely differentiable	infinitely differentiable
$\Pi_{prat}(\vec{f}, D)$	rational	rational
$\Pi_{poly}(\vec{f}, D)$	polynomial	continuous
$\Pi_{rpoly}(\vec{f}, D)$	polynomial	rational
$\Pi_{ppoly}(\vec{f}, D)$	polynomial	polynomial
$\Pi^k_{ppoly}(\vec{f}, D)$	homogeneous polynomial of degree $\leq k$	polynomial
$\Pi_{linear}(\vec{f}, D)$	linear	polynomial

Table 1

In the rest of this thesis, we use terms like purely rational protocols (respectively, polynomial protocols, purely polynomial protocols), for protocols of $\Pi_{rat}(\vec{f}; D)$ (respectively, $\Pi_{poly}(\vec{f}; D)$, $\Pi_{ppoly}(\vec{f}; D)$). We use notation like $C_1(\vec{f}; D)$, $C_2(\vec{f}; D)$, etc., to denote the communication complexity under the restrictions on the protocols introduced in Table 1. Notice that as we go down the table additional restrictions are introduced and, therefore, the corresponding communication complexity can only increase. Finally, assuming that D has nonempty interior, we see that the set $\Pi_{rat}(\vec{f}; D)$ (respectively, $\Pi_{linear}(\vec{f}; D)$) is empty unless $\vec{f}$ is a rational (respectively, polynomial) function.

All of our definitions can be extended, in the obvious way, to the case where the real number field $\Re$ is replaced by the complex field C. Here, all the functions f_i are defined on a subset $D_{\vec{f}}$ of $C^m \times C^n$ and take values in C. Furthermore, a protocol has a domain $D \subset C^m \times C^n$ and the message functions m_i and $\hat{m}_i$ [cf. Eqs.

(1.1.1)–(1.1.2)] are defined on D and are complex–valued.

1.1.2 Communication Protocols, Discrete Model

In this subsection, we describe two kinds of discrete communication protocols whose messages are finite binary sequences. The first one is a discrete version of the continuous model introduced in the previous subsection. The second one is a variant of the continuous set up which provides a framework for studying the communication complexity of inexact algebraic computations (within some desired accuracy). Here, the objective is to minimize the number of binary messages, as a function of the desired accuracy of the final result.

Discrete Communication Protocols

Let $f_1(x,y),\ldots,f_s(x,y)$ be a set of Boolean functions, where $x \in \{0,1\}^m$ and $y \in \{0,1\}^n$. We consider a situation where two processors P_1 and P_2 are to compute $f_1,\ldots,f_s$ under the assumption that processor P_1 (respectively, P_2) has access to the Boolean variable x (respectively, y) only. In addition, the processors possess the truth tables of functions $f_1,\ldots,f_s$. The processors are to exchange a number of binary messages, according to some protocol, until they find values of $f_1,\ldots,f_s$. Our objective is to determine protocols under which the number of exchanged messages is minimized.

A discrete communication protocol π for computing $f_1,\ldots,f_s$ is defined in the same way as a continuous communication protocol (see Subsection 1.1.1), except that the vectors x and y are binary vectors instead of real vectors and that the functions $\hat{m}_i$ and h_j are Boolean functions instead of real valued functions. There is another difference between the discrete and the continuous protocols, namely, no smoothness conditions are imposed on the functions $\hat{m}_i$ and h_j ($i = 1,\ldots,r(\pi)$ and $j = 1,\ldots,s$). As we shall see later in Chapter 4, the smoothness conditions are automatically satisfied if we view each Boolean function f as polynomial on the field $\mathcal{F} = \{0,1;\oplus,\otimes\}$, where $\oplus$ and $\otimes$ are the usual modular 2 addition and multiplication operators. Therefore, all the discrete communication protocols can

be viewed as purely polynomial protocols over the field $\mathcal{F}$, which are denoted by $\Pi_{ppoly}(f;\mathcal{F})$ according to the notation given in Table 1.

The notions of one–way and two–way communication complexity for computing $f_1,\ldots,f_s$ are defined in the same way as before (see Eqs. (1.1.5)–(1.1.6)). We will use a simpler notation $C(\vec{f};\leftarrow)$ (respectively, $C(\vec{f};\leftrightarrow)$) to denote the one–way communication complexity (respectively, the two–way communication complexity) for computing $f_1,\ldots,f_s$ over the field $\mathcal{F}$ using discrete protocols.

ϵ–Communication Complexity

We now describe another framework for studying the problem of approximate algebraic computation of real–valued functions. As a variation of the continuous model, it lets the messages be finite binary sequences and requires that the functions be computed only within some desired accuracy.

To be more precise, let ϵ be some real positive number and $f_1(x,y),\ldots,f_s(x,y)$ be a set of real–valued functions defined on some subset $D = D_x \times D_y$ of $\Re^m \times \Re^n$. As usual, we consider a situation where two processors P_1 and P_2 are to compute the functions $f_1,\ldots,f_s$, assuming that each vector x and y is given to a different processor. Then, a two–way communication protocol consists of
a) A termination time T;
b) A collection of functions $\hat{m}_{i,t}$, $i = 1,2$, $t = 0,1,2,...,T-1$, where $\hat{m}_{1,t} : D_x \times \{0,1\}^t \mapsto \{0,1\}$ and $\hat{m}_{2,t} : D_y \times \{0,1\}^t \mapsto \{0,1\}$.

A protocol corresponds to the following sequence of events. Processor P_1 (respectively, processor P_2) receives its "input", namely, the vector x (respectively, y) and the functional forms of functions $f_1,\ldots,f_s$, and then, at each time t, transmits to processor P_2 (respectively, P_1) a binary message $m_{1,t}$ determined by

$$m_{1,t}(x,y) = \hat{m}_{1,t}(x, m_{2,0}(x,y),...,m_{2,t-1}(x,y))$$

(respectively, $m_{2,t}(x,y) = \hat{m}_{2,t}(y, m_{1,0}(x,y),...,m_{1,t-1}(x,y))$). Thus, the message transmitted by processor P_1 is only a function of the value of x known by it, together with all messages it has received in the past. The actions taken by processor P_2 can be described in a symmetrical fashion. At time T the exchange of messages

ceases. We say that π is an *ϵ-legitimate protocol* for computing $f_1, \ldots, f_s$ if each function f_j can be computed by either processor P_1 or processor P_2 at time T within ϵ–accuracy. In other words, π is ϵ–legitimate if for each f_j $(j = 1, \ldots, s)$ at least one of the two following conditions hold:

a) There exists a function $h_j : D_x \times \{0,1\}^T \mapsto \Re$ such that

$$|f_j(x,y) - h_j(x, m_{2,0}(x,y), ..., m_{2,T-1}(x,y))| \leq \epsilon, \qquad \forall (x,y) \in D. \tag{1.1.7}$$

b) There exists a function $h_j : D_y \times \{0,1\}^T \mapsto \Re$ such that

$$|f_j(x,y) - h_j(y, m_{1,0}(x,y), ..., m_{1,T-1}(x,y))| \leq \epsilon, \qquad \forall (x,y) \in D. \tag{1.1.8}$$

The number of messages transmitted under protocol π, denoted by $r(\pi)$, is simply $2T$. We use $\Pi(\vec{f}; \epsilon, D)$ to denote the set of all ϵ–legitimate protocols. Finally, we let

$$C(\vec{f}; \epsilon, D) = \min_{\pi \in \Pi(\vec{f}; \epsilon, D)} r(\pi).$$

The class of one–way communication protocols and the corresponding notion of one–way ϵ–communication complexity can be defined accordingly.

A couple of remarks on our definition of protocols are in order.

(i) We could have defined these protocols in the same way as we defined the protocols for computing Boolean functions. However, for notational convenience, we have chosen to define them in such a way that each processor must transmit exactly one binary message at each stage. This may be wasteful if, for example, a better protocol may be found in which P_1 first sends many messages and then P_2 transmits its own messages. Nevertheless, the waste that results can be at most a factor of two. If one is only interested in orders of magnitude, this issue is unimportant.

(ii) We have assumed that the termination time T is the same for all $(x,y) \in D$, even though for certain "easy" input vectors (x,y) the desired results may have been obtained earlier. Again, this is of no concern, if we are interested in a worst case analysis.

(iii) Occasionally, we will also consider problems of infinite dimension. In other words, $f_1, \ldots, f_s$ are no longer functions defined on a finite dimensional Euclidean

space, but instead, they are some functionals defined on a subset of some product space $\mathcal{B}_1 \times \mathcal{B}_2$ and take values in $\Re$. For example, we can let Θ be a set of convex functions defined on the n–dimensional bounded domain $[0,1]^n$ and f be a functional defined as

$$f(g_1, g_2) = \min_{x \in [0,1]^n} (g_1(x) + g_2(x)),$$

where g_1, g_2, by a slight abuse of notation, are inputs taken from Θ. (Following the notations used before, g_1 and g_2 should have been denoted as x and y respectively.) Typically, Θ will be defined by imposing certain smoothness conditions on its elements. In light of equations (1.1.7) and (1.1.8), a protocol π belongs to $\Pi(f; \epsilon, \Theta \times \Theta)$ if and only if the final output $h(g_1, m_{2,0}(g_1, g_2), ..., m_{2,T-1}(g_1, g_2))$ is an ϵ–optimal value of the convex program:

$$\min_{x \in [0,1]^n} (g_1(x) + g_2(x)),$$

where the $m_{2,i}$'s are messages sent by processor P_2. (We have assumed that processor P_1 performs the final evaluation.) In other words, there holds

$$h(g_1, m_{2,0}(g_1, g_2), ..., m_{2,T-1}(g_1, g_2)) \leq g_1(x) + g_2(x) + \epsilon,$$

for all $x \in [0,1]^n$ and for all $g_1, g_2 \in \Theta$.

1.2 Literature Survey

In what follows we make an attempt to survey the most important advances in the study of communication complexity.

1.2.1 Continuous Case

The problem formulation for the continuous case, described in Section 1.1, is due to Abelson ([A 78], [A 80]) who established lower bounds on one–way and two–way communication complexity, assuming that the message functions are once (respectively, twice) continuously differentiable. (These results are stated and discussed in Chapter 2 and Chapter 3, respectively.)

Notice that each protocol, as defined in Section 1.1.1, gives rise to a representation of $f(x,y)$, the function to be computed, as a composition of the message functions. These compositions are of a special form, namely, they satisfy equations (1.1.1)–(1.1.4). A similar problem involving representability of continuous functions of several variables by compositions of functions of one variable and by sums of functions was studied by the Russian mathematician A.N. Kolmogorov. Notice that examples of genuine functions of two variables are rare. Of course, $x+y$ is one such function. But all other functions can be reduced to this trivial one by means of functions of one variable. As an example, we see that $xy = e^{\log x + \log y}$.

Theorem 1.2.1 [Kolmogorov] *There exist n constants $0 < \lambda_p \leq 1$, $p = 1,\ldots,n$, and $2n+1$ increasing functions $\phi_q : [0,1] \mapsto [0,1]$, $q = 0,\ldots,2n$, such that for each continuous function f on $[0,1]^n$, there exists a continuous function $g : [0,n] \mapsto \Re$ with the property*

$$f(x_1,\ldots,x_n) = \sum_{q=0}^{2n} g\left(\lambda_1\phi_q(x_1) + \cdots + \lambda_n\phi_q(x_n)\right).$$

The above formula reduces the function f to sums and compositions of g and ϕ_q, $q = 0,\ldots,2n$. Moreover, Theorem 1.2.1 says that the constants λ_p and the functions ϕ_q ($p = 1,\ldots,n$ and $q = 0,\ldots,2n$) do not depend on the function f. The history of Theorem 1.2.1 stems from Hilbert's thirteenth problem which contained (implicitly) the conjecture that not all continuous functions of three variables can be written as compositions of continuous functions of two variables. This conjecture is clearly refuted by the above remarkable theorem of Kolmogorov (See Chapter 11 of [L 66] for a more detailed account.).

1.2.2 Discrete Case

The formulation of discrete communication protocols given in Section 1.1.2 is due to Yao ([Y 79]), who considered the problem of computing Boolean functions using binary messages. His approach is combinatorial in nature and very different from that of Abelson's. The following definition was introduced in [Y 79].

Definition 1.2.1 *Let f be a Boolean function on $\{0,1\}^m \times \{0,1\}^n$. We shall call a Cartesian product $S \times T$ (where $S \subseteq \{0,1\}^m, T \subseteq \{0,1\}^n$) an f-monochromatic rectangle if f is constant over $S \times T$. A k-decomposition of f is a family $\Omega = \{S_1 \times T_1, S_2 \times T_2, \ldots, S_k \times T_k\}$ of k disjoint f-monochromatic rectangles that partition $\{0,1\}^m \times \{0,1\}^n$. Let $d(f)$ be the minimum k such that a k-decomposition of f exists.*

The following basic result is due to Yao ([Y 79]).

Theorem 1.2.2 **[Y 79]** *For any Boolean function f defined on $\{0,1\}^m \times \{0,1\}^n$, there holds*

$$\log_2 d(f) - 2 \leq C(f; \rightarrow) \leq O\left(\sqrt{d(f) \log_2 d(f)}\right).$$

Communication complexity has been studied under a slightly different setting (e.g. [PS 82], [JKS 84] and [DGS 84]) where the objective is to determine the smallest number of 0–1 bits that have to be exchanged for computing a Boolean function $f(x)$, minimized over all partitions of the input in two equal parts. It was shown that for most Boolean functions $f(x)$ defined on $\{0,1\}^n$ the communication complexity (defined in the new sense) is $O(n)$.

The new notion of communication complexity ([MS 82], [PS 82]) was motivated, in part, by the problem of finding the optimal design of a VLSI chip. Several papers ([AU 83], [JK 84], [LS], [U 84]), including [MS 82] and[PS 82], have pointed out that communication complexity (in the sense of [PS 82]) of a Boolean function f provides a direct lower bound on the minimum bisection width ([T 79]) of any chip that computes f. In [AU 83], several known lower bound techniques are compared and a new lower bound approach based on a better adversary argument has been found. In addition, [AU 83] has given a new upper bound for the two–way communication complexity, namely,

$$C(f; \leftrightarrow) \leq O(\log^2 d(f)),$$

which impoves upon Yao's result (Theorem 1.2.2).

The notion of communication complexity can also be formulated in the context of probabilistic computation. Indeed, an interesting recent innovation in algorithm

design is the inclusion of randomness that allows a solution to be incorrect with probability less than some threshold level ([R 76]). Suppose that there are two processors P_1 and P_2 who wish to determine the value of some Boolean function $f(x,y)$, under the assumption that processor P_1 (respectively, processor P_2) knows only the Boolean variable x (respectively, y) and the truth table of f. A one–way probabilistic communication protocol π for computing f consists of $2^m \times 2^n$ pairs of probability distributions $P(x) = (p_1(x), p_2(x), \ldots, p_K(x))$, and $Q(y) = (q_1(y), q_2(y), \ldots, q_K(y))$, for each $(x,y) \in \{0,1\}^m \times \{0,1\}^n$, where K is some positive integer. These probability distributions have to satisfy the following conditions:

$$\begin{cases} \sum_l p_l(x) = 1, & \forall x; \\ p_l(x) \geq 0, \quad 0 \leq q_l(y) \leq 1, & \forall x, y, l; \\ P(x)Q(y) = \sum_l p_l(x) q_l(y) \geq (1-\epsilon), & \text{if } f(x,y) = 1; \\ P(x)Q(y) = \sum_l p_l(x) q_l(y) < \epsilon, & \text{if } f(x,y) = 0. \end{cases} \tag{1.2.1}$$

The actual computation of f using the protocol π goes as follows: for each input vector $(x,y) \in \{0,1\}^m \times \{0,1\}^n$, processor P_1 sends stochastically to processor P_2 a binary string of length K, say, $a_1 a_2 \ldots a_K$, such that $\text{Prob}(a_l = 1) = p_l(x)$ and $\text{Prob}(a_l = 0) = 1 - p_l(x)$, $l = 1, \ldots, K$. After receiving the string, assuming that no noise or distortion is present in the information transmitting channels, processor P_2 decides the value of $f(x,y)$ using the rule:

$$\begin{cases} f(x,y) = 1, & \text{if } \sum_{l=1}^{K} a_l q_l(y) \geq \frac{1}{2}; \\ f(x,y) = 0, & \text{if } \sum_{l=1}^{K} a_l q_l(y) < \frac{1}{2}. \end{cases}$$

In light of condition (1.2.1), we see that the expected value of $\sum_{l=1}^{K} a_l q_l(y)$ is less than ϵ if $f(x,y) = 0$ and is greater than $1-\epsilon$ if $f(x,y) = 1$. Thus, the chance of error of protocol π can be made arbitrarily small by adjusting the value of ϵ. The one–way probabilistic communication complexity for computing a Boolean function $f(x,y)$ is then defined to be $\lceil \log_2 K \rceil$, where K is the minimum integer such that there exists vectors $P(x) = (p_1(x), p_2(x), \ldots, p_K(x))$, and $Q(y) = (q_1(y), q_2(y), \ldots, q_K(y))$, $\forall x \in \{0,1\}^m$ and $\forall y \in \{0,1\}^n$, satisfying the following relation (1.2.1). One can also define the notion of two-way probabilistic communication complexity in a similar manner. Several complexity results under such models were obtained by Yao

([Y 79]). For example, it has been shown that with probability at least $(1-O(2^{\frac{-n^2}{2}}))$ a random Boolean function f with an $n \times n$ truth table satisfies

$$C(f;\epsilon) \geq \lceil \log_2 n \rceil - \log_2 \log_2 n - 2,$$

where $0 < \epsilon < \frac{1}{2}$ and $C(f;\epsilon)$ denotes the two–way probabilistic communication complexity.

Similar to the probabilistic protocols, the so called Las Vegas protocols also make use of internal randomizations. However, the output of a Las Vegas protocol does not depend on the random events during message transmission, that is, a Las Vegas protocol has zero probability of error. The number of messages used by a Las Vegas protocol is the expected number of messages being sent, averaged over all possible outcomes of the randomizations. It was shown ([MS 82]) that Las Vegas protocols are more powerful than deterministic protocols. On the other hand, [AU 83] has shown that the deterministic communication complexity of any Boolean function cannot be more than the square of the Las Vegas communication complexity. An example was given in [F 87] whose deterministic communication complexity is exactly equal to the square of its Las Vegas communication complexity.

Apart from *randomness*, *nondeterminism* can also be introduced to the study of communication complexity([PS 82]). In particular, a nondeterministic protocol can be defined naturally through the model in Section 1.1.2 by simply allowing each $\hat{m}_i$ to be a point to set mapping. As a result, the value of $h_j(x,y)$, which depends on the values of $\hat{m}_j$'s to be chosen, is not unique. The final output of the computation is defined to be 1 if and only if there exists a possible computation such that $h_j(x,y) = 1$. It was shown in [PS 82] that nondeterminism can provably provide exponential savings over deterministic protocols. It was also shown that two–way communication protocols can be exponentially more powerful than one–way communication protocols. In Chapter 4, we will give another proof of this fact.

[HR 88] has compared the communication complexity under several different models of communication (deterministic, nondeterministic, probabilistic, one–way as well as two–way).

A fair amount of research has dealt with extensions of the results of [Y 79] and with the evaluation of the communication complexity of selected combinatorial problems (e.g. [JK 84], [PE 86]). For example, in [JK 84], Ja'Ja' has obtained several lower bounds on the communication complexity of certain graph problems. It was shown that given an undirected graph $G = (V, E)$ with n nodes, if the lower triangular part of the adjacency matrix of G is partitioned into two equal parts between two processors P_1, P_2, then at least $\Omega(n \log n)$ bits of communication are needed for the two processors to determine the connected components of G. This bound is tight since there exists an obvious protocol that accomplishes this end and communicates at most $O(n \log n)$ bits of information. In fact, we can simply let processor P_1 send to processor P_2 all its information. It is easily seen that at most $O(n \log n)$ bits are needed to code the information possessed by processor P_1. Similar results were also obtained for other graph problems such as determining the biconnected components, the bridge–connected components, the strongly connected components, a minimum weight spanning tree, and the transitive closure. As mentioned before, all of these results can be translated into lower bounds for the VLSI area–time complexity for solving the corresponding graph problems. The work of [HMT 88] also dealt with the communication complexity of graph properties. But their approach is more algebraic in nature. In particular, they proved an $\Omega(n \log n)$ lower bound for the graph connectivity problem (the decision version of the connected component problem) for which the method of [JK 84] does not seem to work.

[MS 82] contained a new lower bound on the deterministic two–way communication complexity under the discrete model described in Section 1.1.2. Their result is very similar to Abelson's result under the continuous set–up. In particular, it was shown that the communication complexity of computing a Boolean function $f(x, y)$, where $x \in \{0,1\}^m$, $y \in \{0,1\}^n$, deterministically with two processors, under the assumption that one processor is given the vector x while the other is given y, is at least $\log_2 \text{rank}(f)$. Here $\text{rank}(f)$ denotes the rank of the truth table of f (which is a 0–1 matrix) over an arbitrary field. Then this lower bound was applied to show that randomized algorithms (Las Vegas algorithm) can be strictly more powerful than deterministic algorithms.

In [JKS 84], lower bound techniques similar to that of [Y 79] are developed for proving lower bounds for the minimum amount of information exchange required in distributed computation. Tight bounds are shown for basic arithmetic and numerical problems such as integer multiplication, matrix squaring, matrix inversion.

Up to now, we have surveyed various lower and upper bounds for the two party communication complexity. Interestingly, no effective general procedure has been found for determining the communication complexity and providing a protocol that achieves the optimality. It turns out that there is a good reason behind it, as Papadimitriou and Tsitsiklis have shown [PT 82] that the problem of determining the communication complexity of a general Boolean function is NP–hard.

It is hard to formulate the problem of communication complexity for a general network consisting of more than two processors, as indicated by the limited literature available on this subject. In all existing work in this direction, either some restrictions were placed on the communication scheme (e.g. [?]) or the network was assumed to have a special structure (e.g. [VB 81], [Ti 84]). For example, [VB 81] studies the communication time required for each processor to send a packet to some other processor in a Butterfly network, and has shown how randomization can used to improve communication. In [Ti 84], it was shown that for almost all Boolean functions f, the communication complexity of determining f on a linear array with $p+1$ processors is approximately p times its communication complexity on a system with two processors. The work of [?] deals with a set of processors who communicate in a special way: each processor has access to the information held by all the other processors except the information held by itself.

The framework given in Section 1.1.2 was introduced in [TL 87] for the problem of approximately minimizing (within a desired accuracy) the sum of two convex functions, with each function known to a different processor. Here, the objective is to minimize the number of binary messages, as a function of the desired accuracy of the solution. The problem of approximating a functional defined on some space (e.g. the integration functional defined on the space of functions integrable over $[0,1]^n$) by arithmetic operations $(+,-,\times,\div)$ was studied in [TW 80] and [TWW 83]. There, the objective is to minimize the number of arithmetic operations used to approximate a given functional within a certain desired accuracy.

Finally, we note that communication complexity, as defined in Section 1.1, does not take into account the computation time needed for generating the messages and computing the final outputs. One reason for this simplification is that in some situations the computation time is negligible as compared to the time used in communication. Some limited work ([PU 84], [PY 88]) has been done in which both the computation and the communication time are taken into consideration. There, the question is how to schedule tasks on a parallel computer consisting of a set of processors, assuming that communication among each pair of processors takes a fixed amount of time. The tasks are given *a priori* as the set of nodes in some directed acyclic graph, unlike the models described in Section 1.1 where the procedure for computing the final outputs depends on the communication protocols being used.

1.3 Outline of the Thesis

The rest of this thesis is organized as follows. In Chapter 2, we study the one–way communication complexity of computing a set $f_1, \ldots, f_s$ of polynomials. The results of [A 78] (stated in Subsection 2.2) provide a complete solution for the case of a single function f, smooth message functions, and protocols whose domain is a (possibly very small) open set. We extend these results to the case of $s > 1$. We also show that we can restrict to the class of polynomial protocols while increasing the communication complexity by at most one. Furthermore, the polynomial protocols we construct have a domain which is almost all of $\Re^m \times \Re^n$ (except for a set of measure zero). We also consider the special case where $m = n$ and each one of the polynomials $f_i : \Re^n \times \Re^n$ is of the form $f_i(x, y) = \hat{f}_i(x + y)$, where each $\hat{f}_i$ is a polynomial in n variables. For this case, we obtain a complete characterization of the communication complexity, a proof that linear protocols are optimal, and a constructive procedure for designing such protocols. Chapter 2 also contains some background results from field extension theory that are used for our study of one–way communication complexity.

In Chapter 3, we derive several general lower bounds on the two–way communication complexity of computing a rational function f when the messages are constrained to be rational functions of the data. Our results are obtained by combin-

ing an earlier result of Abelson [A 80] with Hilbert's Nullstellensatz from algebraic geometry. We also identify certain instances where the lower bounds of [A 80] are tight. Then, these results are applied to the problem of computing a particular entry of the inverse of $x+y$, where x and y are $n \times n$ complex matrices. We derive an n^2-1 lower bound (which agrees with the obvious upper bound, within one message), while the results of [A80] could only provide an $\Omega(n)$ lower bound. We also develop a new lower bound that makes use of first order derivatives of f, in contrast to Abelson's result which uses the second order derivatives of f. We then appy this new lower bound to the problem where f is given as a root z of the polynomial $\sum_{i=0}^{n-1}(x_i+y_i)z^i$, and obtain a lower bound n, as opposed to the $\Omega(1)$ lower bound obtained by applying Abelson's result. We also consider the problem of computing multiple functions and examine some of the related combinatorial issues. For the special case where the functions to be computed are simple quadratic polynomials, we show that the problem of designing optimal protocols can be converted to the problem of finding a minimum node cover for certain bipartite graph, which is solvable in polynomial time using bipartite matching algorithms.

In Chapter 4, we study the discrete communication complexity. We first point out that Abelson's lower bound for the two–way continuous protocols can be applied to discrete protocols, provided that we view each Boolean function as a polynomial over an appropriately defined field. Then, we consider a situation where two processors P_1 and P_2 are to approximately minimize the sum of two convex functions f_1+f_2, under the assumption that each processor P_i has access to the function f_i $(i=1,2)$ only. We provide both upper and lower bounds on the number of bits that have to be exchanged between the two processors in order to find an ϵ–optimal solution. We then apply these results and a result of Dudley ([D 74]) to the problem of deciding whether two given convex sets are disjoint and show that two–way communication protocols can provide exponential savings over one–way communication protocols. (This result was first proved by Papadimitriou and Sipser [PS 82].)

In Chapter 5, we consider multi–party communication protocols in which more than two processors exchange information in order to compute a given function. Two kinds of communication protocols are considered, namely, the protocols in which messages are real–valued functions (continuous protocols) and those in which

the messages are binary strings (discrete protocols). We demonstrate, by using computational complexity theory, that the problem of designing optimal continuous multi–party protocols for computing simple functions such as the linear functions is combinatorially intractable. Then, we consider a class of multi–party protocols in which the communication channels have a tree structure and the messages are binary strings. For such class of protocols, we show almost tight lower bounds on the communication complexity of computing simple functions such as $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i$, or $f(x_1, \ldots, x_n) = \prod_{i=1}^{n} x_i$, up to some prespecified accuracy ϵ.

In Chapter 6 we summarize the conclusions and suggest general directions for future research. Finally, Appendix A contains some results about multi–variable calculus which are used in Chapter 3.

Chapter 2

One–Way Communication Complexity

In this chapter, we give an almost optimal lower bound for the one–way communication complexity of computing a set of multivariable polynomials with protocols whose message functions are polynomials. Our result generalizes a result of Abelson's which deals with the problem of computing only one function. Our proof is constructive and makes use of several results from field extension theory, in contrast to the proof of Abelson result which is local and nonconstructive because it is based on the implicit function theorem. We also consider a special case where each of the polynomials to be computed is of a special form, namely, $f_j(x,y) = \hat{f}_j(x+y)$, for some polynomial $\hat{f}_j$. We show that for such problems there exist optimal protocols with a very simple structure: they consist of messages which are linear functions of the data. The rest of this chapter is organized as follows. Section 2.1 contains some background material from field extension theory. In Section 2.2, we prove the major lower bound result by using results of Section 2.1. Finally, Section 2.2.2 is devoted to the problem of computing polynomials of the form: $f_j(x,y) = \hat{f}_j(x+y)$, $j = 1, \ldots, s$.

2.1 Field Extension Theory

In this section, we introduce some algebraic results (see e.g. [ZS 65, pages 95-125] or [VW 53]) from field extension theory that will be needed in Section 2.2.

Notation: Let $\{a_i : i \in I\}$ be a collection of vectors in $\Re^n$, where I is a finite index

set. We use $[a_i : i \in I]$ to denote the matrix with columns a_i, $i \in I$. Whenever the range of the index variable i (that is, the index set I) is evident from the context, we use the simpler notation $[a_i : i]$. For any function $f : \Re^n \mapsto \Re$, we use ∇f to denote the vector–valued function whose components are the partial derivatives of f. We also use $\nabla f(p)$ to denote the value of ∇f evaluated at some $p \in \Re^n$.

Definition 2.1.1 *Let F_1 and F_2 be two fields. We say that F_2 is an extension of F_1, denoted by F_2/F_1, if F_1 is a subfield of F_2. An element $\lambda \in F_2$ is said to be algebraic over F_1 if λ satisfies a relation $f(\lambda) = 0$, where f is a polynomial with coefficients taken from F_1. We say that F_2 is an algebraic extension field of F_1 if all the elements of F_2 are algebraic over F_1. Otherwise, we say that F_2 is a transcendental extension field of F_1.*

Let F_1 be a subfield of some field F. A typical way of constructing an extension field of F_1 is by adjoining to F_1 some elements $\lambda_i \in F$ that do not belong to F_1 (i in some index set $\mathcal{A}$). Consider the set of all subfields of F that contain F_1 and λ_i ($i \in \mathcal{A}$). The intersection of all of these fields is still a field and is the smallest field containing F_1 and the λ_i's. This field is called the field generated by the λ_i's and will be denoted by $F_2 = F_1(\{\lambda_i, i \in \mathcal{A}\})$. When the cardinality of $\mathcal{A}$ is finite, we say that F_2 is a finitely generated extension field of F_1.

Definition 2.1.2 *We say that F_2 is a finite algebraic extension of the field F_1, if the extension F_2/F_1 is algebraic and the dimension of F_2, when regarded as a vector space over F_1, is finite.*

Definition 2.1.3 *Let F_2/F_1 be a finite algebraic extension and let λ be an element of F_2. We say that λ is a primitive element of the extension F_2/F_1 if $F_2 = F_1(\lambda)$, i.e., if F_2 is generated by λ over the field F_1. In this case, we say that F_2 is a simple extension of F_1.*

The notion of a finite algebraic extension is different from the notion of a finitely extension generated. For example, $\Re(x)$ is a finitely generated field over $\Re$ but not a finite algebraic extension since $\Re(x)/\Re$ is a transcendental extension. However, the

following theorem states that this is the only type of counterexample (see [ZS 65, pages 60–61]).

Theorem 2.1.1 *Every finitely generated algebraic extension is finite.*

Definition 2.1.4 *Let F be a field and let $F[x]$ be the ring of polynomials with coefficients in F. The differentiation operator $\frac{d}{dx}$ is the mapping of $F[x]$ into itself defined in terms of the following properties:*

$$\frac{d}{dx}\left(\sum_{i=0}^{n} a_i x^i\right) = \sum_{i=1}^{n} i a_i x^{i-1},$$

where $n \geq 0$ and $a_i \in F$, $i = 0, \ldots, n$.

We note that when F is equal to $\Re$, the above definition coincides with the usual notion of differentiation.

Definition 2.1.5 *Let F_2/F_1 be an algebraic extension and let λ be an element in F_2. The minimal polynomial of λ is a polynomial $f \in F_1[x]$ of the smallest degree such that $f(\lambda) = 0$. The element λ is called separable over F_1 if there holds $\left(\frac{d}{dx}(f)\right)(\lambda) \neq 0$, where f is the minimal polynomial of λ. F_2 is called a separable algebraic extension if all the elements of F_2 are separable over F_1.*

The following result (see e.g. [ZS 65, page 84]) is called the theorem of primitive element and will be used in Section 2.2.

Theorem 2.1.2 *Every finite separable algebraic extension F_2/F_1 has a primitive element. Hence, every such extension is a simple extension. Furthermore, if $F_2 = F_1(\lambda_1, \ldots, \lambda_k)$, then there exists a primitive element of the form $\lambda = \sum_{j=1}^{k} \gamma_j \lambda_j$ where $\gamma_j \in F_1$ for each j.*

Remark: In fact, the proof of Theorem 2.1.2 given in [ZS 65, page 84] shows that a primitive element λ is obtained for an arbitrary choice of the coefficients $\gamma_1, \ldots, \gamma_k$ as long as they do not lie in the zero set of a certain polynomial. As an illustration, notice that $\Re(1+i, 1-i) = C$ is a finite separable algebraic extension over $\Re$.

By Theorem 2.1.2, there exists a primitive element which can be taken as a linear combination of $1+i, 1-i$. In particular, one has $\Re(1+i, 1-i) = \Re(\gamma_1(1+i)+\gamma_2(1-i))$ for some suitable choices of real numbers γ_1, γ_2. It is not hard to see that all of the fields $\Re(\gamma_1(1+i)+\gamma_2(1-i))$ are equal to C, as long as $\gamma_1 \neq \gamma_2$.

We now turn our attention to the case of transcendental extensions.

Definition 2.1.6 *Let F_2/F_1 be a field extension. The transcendental degree of F_2/F_1 is defined as the smallest number t such that there exist elements $\lambda_1, \lambda_2, \ldots, \lambda_t$ in F_2 with the property that F_2 is an algebraic extension of $F_1(\lambda_1, \lambda_2, \ldots, \lambda_t)$. In particular, if F_2/F_1 is an algebraic extension to start with, then its transcendental extension degree is zero. The transcendental degree will be denoted by* tr.d.F_2/F_1 *and the elements $\lambda_1, \lambda_2, \ldots, \lambda_t$ will be called a transcendental basis of F_2/F_1.*

In light of the above definition, $\Re(x_1, x_2, \ldots, x_m)$ (the field of rational functions over $\Re$ with indeterminates $x_1, x_2, \ldots, x_m$) is a transcendental extension of $\Re$ with degree m and $x_1, x_2, \ldots, x_m$ can be taken as a transcendental basis. The following theorem summarizes some important properties of the transcendental degree of a field extension.

Theorem 2.1.3 *Let F_2 be a finitely generated extension field of F_1 and let F_3 be a finitely generated extension field of F_2. (In particular, F_3 is also a finitely generated extension field of F_1.) Suppose that $F_3 = F_1(\lambda_1, \lambda_2, \ldots, \lambda_n)$ and that* tr.d.$F_3/F_1 = t$. *Then,*

$$t = \text{tr.d.}F_3/F_1 = \text{tr.d.}F_3/F_2 + \text{tr.d.}F_2/F_1.$$

The following is the definition of a derivation over a field, which is a generalized notion of differentiation.

Definition 2.1.7 *Let F_2 be a finitely generated extension field of F_1 and let F_3 be an extension field of F_2. A mapping D of F_2 into F_3 is said to be an F_1-derivation of F_2 (with values in F_3) if, for every λ in F_1 and every x, y in F_2, the mapping D has the following properties:*

1. $D(\lambda) = 0$;

2. $D(x + y) = D(x) + D(y)$;

3. $D(xy) = xD(y) + yD(x)$.

Notice that the derivations are defined in a way that is very similar to differentiations. As a result, one can show that the well known chain rules remain true for derivations. We now let $\mathcal{D}_{F_2/F_1}(F_3)$ stand for the space of all F_1–derivations of F_2 with values in F_3. Then $\mathcal{D}_{F_2/F_1}(F_3)$ can be viewed as a vector space over F_3 in a natural way since one can easily verify that $\mathcal{D}_{F_2/F_1}(F_3)$ is closed under linear combinations over F_3. It can be shown (see [ZS 65, pages120–127]) that the dimension of the vector space $\mathcal{D}_{F_2/F_1}(F_3)$ does not depend on the particular choice of F_3. It is for this reason that we usually drop F_3 from the notation $\mathcal{D}_{F_2/F_1}(F_3)$ and use simply $\mathcal{D}_{F_2/F_1}$ to denote the space of F_1–derivations of F_2 with values in any extension field of F_2.

Definition 2.1.8 *Let F be a field whose multiplication identity is denoted by e. If $\sum_{i=1}^{n} e \neq 0$ for all positive integers n, we say that F has characteristic* 0.

For example, the fields $\Re$ and C have characteristic 0. In fact, every extension field of $\Re$ has characteristic 0 since it shares the same identity element with $\Re$. The following result is quoted from [ZS 65, page 125].

Theorem 2.1.4 *Let F_2 be a finitely generated extension field of F_1 and let F_3 be a finitely generated extension field of F_2. If F_2 has characteristic zero, then each derivation $D \in \mathcal{D}_{F_2/F_1}$ can be extended to a derivation $\overline{D}$ in $\mathcal{D}_{F_3/F_1}$.*

Example: We now consider in some detail the space of derivations for an important special case and derive a result that will be needed in Section 2.2. Let $F_1 = \Re$ and let $F_3 = \Re(x_1, x_2, \ldots x_m)$, the field of rational functions over $\Re$ with indeterminates $x_1, x_2, \ldots, x_m$. Furthermore, we let F_2 be the subfield of F_3 which is generated by certain polynomials $f_1, f_2, \ldots, f_n \in F_3$. In other words, F_2 is the set of all rational

functions that can be expressed as rational functions of the f_j's. It can be readily verified that the partial derivatives $\frac{\partial}{\partial x_k}$, defined by

$$\left(\frac{\partial}{\partial x_k}\right)(x_j) = \delta_{jk},$$

are in $\mathcal{D}_{F_3/F_1}(F_3)$, where δ_{jk} is the Kronecker delta. This implies that for any $D \in \mathcal{D}_{F_3/F_1}(F_3)$ the derivation $(D - \sum_{k=1}^{m} D(x_k)\frac{\partial}{\partial x_k})$ maps $x_1, \ldots, x_m$ to zero. Hence it maps F_3 to zero. In other words, we have

$$D = \sum_{k=1}^{m} D(x_k)\frac{\partial}{\partial x_k}.$$

Hence D is completely determined by the choices of $D(x_k) \in F_3$, $k = 1, 2, \ldots, m$, and $\{\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_m}\}$ is a basis for $\mathcal{D}_{F_3/F_1}(F_3)$. Now suppose that $D \in \mathcal{D}_{F_2/F_1}(F_3)$. Since F_2 has characteristic 0, by Theorem 2.1.4, we see that D can extended to a derivation $\overline{D}$ in $\mathcal{D}_{F_3/F_1}(F_3)$. From the above discussion, we see that

$$\overline{D} = \sum_{k=1}^{m} \overline{D}(x_k)\frac{\partial}{\partial x_k}. \tag{2.1.1}$$

Therefore, the map D, which is equal to the restriction of $\overline{D}$ on F_2, can be written as a linear combination of the $\frac{\partial}{\partial x_k}$'s (cf. Eq. (2.1.1)). Conversely, for each choice of $\overline{D}(x_k) \in F_3$, Eq. (2.1.1) defines a derivation in $\mathcal{D}_{F_2/F_1}(F_3)$. However, two different choices of $\overline{D}(x_k)$ may give rise to the same derivation in $\mathcal{D}_{F_2/F_1}(F_3)$. As a matter of fact, any $f \in F_2$ can expressed in the form of $f = g(f_1, f_2, \ldots, f_n)$, where $g(z_1, z_2, \ldots z_n)$ is a rational function. By the chain rule, we have

$$D(f) = \frac{\partial g}{\partial z_1}D(f_1) + \frac{\partial g}{\partial z_2}D(f_2) + \cdots + \frac{\partial g}{\partial z_n}D(f_n),$$

where $\frac{\partial g}{\partial z_j}$ is the partial derivative of g with respect to z_j defined in the usual sense. Since the $\frac{\partial g}{\partial z_j}$'s are independent of D, we see that D is completely determined by its operation on f_j, $j = 1, 2, \ldots, n$. Moreover, since the f_j's belong to F_2 we see that different choices of the $D(f_j)$'s will result in different derivations in $\mathcal{D}_{F_2/F_1}$.

We now develop an explicit formula for the dimension of $\mathcal{D}_{F_2/F_1}$ (Eq. (2.1.4) below), in the context of the particular example we have been considering. This formula will be crucial for the results of Section 2.2.

Notice that for every j and any $D \in \mathcal{D}_{F_2/F_1}$, one has

$$\begin{aligned} D(f_j) &= \left(\sum_{k=1}^{m} \overline{D}(x_k)\frac{\partial}{\partial x_k}\right)(f_j) \\ &= \sum_{k=1}^{m} \overline{D}(x_k)\frac{\partial f_j}{\partial x_k} \\ &= \left(\overline{D}(x_1), \overline{D}(x_2), \ldots, \overline{D}(x_m)\right) \nabla f_j. \end{aligned} \tag{2.1.2}$$

We now rewrite Eq. (2.1.2) in the matrix form

$$(D(f_1), D(f_2), \ldots, D(f_n)) = \left(\overline{D}(x_1), \overline{D}(x_2), \ldots, \overline{D}(x_m)\right) [\nabla f_j : j \in J],$$

where $J = \{1, 2, \ldots, n\}$. Since $\overline{D}(x_k)$ can be taken arbitrarily, we see that the vector space $\mathcal{D}_{F_2/F_1}(F_3)$ is isomorphic to the space spanned by the rows of the matrix $[\nabla f_j : j]$. Hence

$$\dim \mathcal{D}_{F_2/F_1} = \operatorname{rank}[\nabla f_j : j], \tag{2.1.3}$$

where the entries of $[\nabla f_j : j]$ are polynomials in the variables $x_1, x_2, \ldots, x_m$ and the rank is evaluated in the field F_3.

An alternative formula for $\dim \mathcal{D}_{F_2/F_1}$ is obtained as follows. We can assign real values to $x_1, x_2, \ldots, x_m$ and calculate the rank in $\Re$. Consider the matrix $[\nabla f_j(p) : j]$ which is the matrix $[\nabla f_j : j]$ evaluated at the point $p \in \Re^m$. Suppose that the maximum (over all p) rank of $[\nabla f_j(p) : j]$ is r. Then there must exist some $p \in \Re^m$ and some submatrix of $[\nabla f_j(p) : j]$ of dimensions $r \times r$ whose determinant is nonzero. Consider the determinant (in F_3) of the corresponding submatrix of $[\nabla f_j : j]$. This determinant is a polynomial which, according to the above discussion, does not vanish at p. Therefore, this determinant is a nonzero polynomial. Consequently, the rank of $[\nabla f_j : j]$ (viewed as a matrix of elements of F_3) is greater than or equal to r. By reversing this argument, we also see that r is no less than the rank of $[\nabla f_j : j]$. Hence

$$\max_{p \in \Re^m} \operatorname{rank}([\nabla f_j(p) : j]) = \operatorname{rank}([\nabla f_j : j]).$$

Combining this with Eq. (2.1.3), we obtain the following basic result:

$$\dim \mathcal{D}_{F_2/F_1} = \max_{p \in \Re^m} \operatorname{rank}([\nabla f_j(p) : j]). \tag{2.1.4}$$

We close this section with a result which relates the transcendental extension degree and the dimension of the associated space of derivations (see [ZS 65, page 125-127]).

Theorem 2.1.5 *Let F_1 be a field and let F_2 be a finitely generated extension field of F_1 such that* tr.d.$F_2/F_1 = d$ *and* $\dim \mathcal{D}_{F_2/F_1} = t$. *Then t is equal to the smallest number r such that there exist elements $\lambda_1, \lambda_2, \ldots, \lambda_r$ with the property that F_2 is separable algebraic over $F_1(\lambda_1, \lambda_2, \ldots, \lambda_r)$. In particular, $t \geq d$. Furthermore, if F_1 has characteristic 0, then the equality $t = d$ holds.*

2.2 One–Way Communication Complexity

In this section, we study the one–way communication complexity of evaluating a set $f_1, \ldots, f_s$ of polynomials, when the messages transmitted are restricted to be polynomial functions of the data. We apply the tools of field extension theory (presented in Section 2.1) to obtain a bound for the communication complexity which is almost optimal (within one message). It will be seen that our results strengthen earlier results in a number of directions. We also show that the restriction to polynomial protocols can increase the communication complexity of the problem by at most one message.

2.2.1 General Results

The main available result on one-way protocols is due to Abelson [A 78]:[1]

Theorem 2.2.1 *Let $f : \Re^m \times \Re^n \mapsto \Re$ be an infinitely differentiable function.*

a) *Let D be a subset of $\Re^m \times \Re^n$. There holds $C_\infty(f; D) \leq r$ if and only if there exist infinitely differentiable functions $m_1, m_2, \ldots, m_r : \Re^n \mapsto \Re$ and*

[1] We state this result for the class $\Pi_\infty(f; D)$ of protocols that use infinitely differentiable functions. The result was actually proved in [A 78] for the class $\Pi_1(f; D)$ but the proof remains valid for $\Pi_\infty(f; D)$.

$h : \Re^{r+n} \mapsto \Re$ such that

$$f(x,y) = h(y, m_1(x), m_2(x), \ldots, m_r(x)), \quad \forall (x,y) \in D. \tag{2.2.1}$$

b) *Let (x^*, y^*) be some element of $\Re^m \times \Re^n$. There exists some open set $D \subset \Re^m \times \Re^n$ containing (x^*, y^*) for which $C_\infty(f; D) \leq r$ if and only if*

$$\dim\left(\text{span}\{g_{1,x^*}, g_{2,x^*}, \ldots, g_{m,x^*}\}\right) \leq r, \tag{2.2.2}$$

where $g_{i,x^*}(y) = \frac{\partial f}{\partial x_i}(x^*, y)$ and where the span is taken in the vector space of functions of y defined on an open set containing y^*.

Let us consider protocols whose domain D is all of $\Re^m \times \Re^n$. By varying (x^*, y^*) over all possible elements of $\Re^m \times \Re^n$ and applying part (b) of the theorem to each one of these points we obtain

$$C_\infty(f; \Re^m \times \Re^n) \geq \max_{x^* \in \Re^m} \dim\left(\text{span}\{g_{1,x^*}, \ldots, g_{m,x^*}\}\right). \tag{2.2.3}$$

Part (b) of the theorem states that this lower bound is also tight in a local sense: there exist protocols whose number of messages equals the lower bound and which evaluate f correctly when (x,y) is restricted to a suitably small domain D. However, nothing can be inferred on the tightness of this bound when one considers protocols whose domain is all of $\Re^m \times \Re^n$. Furthermore, the message functions m_i in Eq. (2.2.1) are not guaranteed to be polynomials, even if the function f is a polynomial. Both of these deficiencies will be remedied in the sequel.

Throughout this section, we assume that we are dealing with a given set $\vec{f} = \{f_1, \ldots, f_s\}$ of polynomial functions mapping $\Re^m \times \Re^n$ into $\Re$ and that only one–way protocols are considered. We start by proving a lower bound similar to Theorem 2.2.1(b), but more general, because Theorem 2.2.1 dealt only with the case $s = 1$.

Notation: For $i = 1, \ldots, s$, and for any set $\alpha = (\alpha_1, \ldots, \alpha_n)$ of nonnegative integer indices, we define a function $g_i^\alpha : \Re^{m+n} \mapsto \Re$ by letting

$$g_i^\alpha(x,y) = \frac{\partial^\alpha f_i}{\partial y_1^{\alpha_1} \partial y_2^{\alpha_2} \cdots \partial y_n^{\alpha_n}}(x,y). \tag{2.2.4}$$

(We use the convention $g_i^{0,\ldots,0} = f_i$.) Let $\mathcal{A}$ be the set of all α such that g_i^α is not identically zero for some i. (Clearly, $\mathcal{A}$ is a finite set, since each f_i is a polynomial.)

For any function $g(x,y) : \Re^m \times \Re^n \mapsto \Re$, we use $\nabla_x g$ to denote the vector–valued function of dimension m whose components are the partial derivatives of g with respect to the first m coordinates.

Theorem 2.2.2 *Let D be some open subset of $\Re^m \times \Re^n$.*

a) *If $C_\infty(\vec{f}; D) \leq r$, then there exist infinitely differentiable functions $m_1, \ldots, m_r : \Re^m \mapsto \Re$ and $h_i^\alpha : \Re^{r+n} \mapsto \Re$, $i = 1, \ldots, s$, $\alpha \in \mathcal{A}$, such that*

$$g_i^\alpha(x,y) = h_i^\alpha(y, m_1(x), \ldots, m_r(x)), \ \forall (x,y) \in D, \ i = 1, \ldots, s. \tag{2.2.5}$$

b) *There holds*

$$C_\infty(\vec{f}; D) \geq \max_{(x,y)\in D} \operatorname{rank}[\nabla_x g_i^\alpha(x,y) : i = 1, 2, \ldots, s; \alpha \in \mathcal{A}]. \tag{2.2.6}$$

Proof: a) Since $C_\infty(\vec{f}; D) \leq r$, there exist infinitely differentiable functions $m_1, \ldots, m_r$ and $g_1, \ldots g_s$ such that

$$f_i(x,y) = h_i(m_1(x), \ldots, m_r(x), y), \quad \forall (x,y) \in D, \ i = 1, \ldots, s.$$

We differentiate both sides of this equation, with respect to y. The left–hand side yields $g_i^\alpha(x,y)$. The right–hand side remains an infinitely differentiable function of $m_j(x)$, $j = 1, \ldots, r$ and y, and h_i^α can be taken equal to that function.

b) Suppose that $C_\infty(\vec{f}; D) = r$. Then Eq. (2.2.5) holds for some suitable functions h_i^α and for all $(x,y) \in D$. By differentiating both sides with respect to x, we obtain

$$\nabla_x g_i^\alpha(x,y) = \sum_{k=1}^{r} \frac{\partial}{\partial m_k} h_i^\alpha(m_1(x), \ldots, m_r(x), y) \cdot \nabla_x m_k(x), \qquad \forall (x,y) \in D, \ \forall i. \tag{2.2.7}$$

Thus, each column of the matrix $[\nabla_x g_i^\alpha(x,y) : i = 1, 2, \ldots, s; \alpha \in \mathcal{A}]$ is a linear combination of the vectors $\nabla_x m_1(x), \ldots, \nabla_x m_r(x)$. It follows that the rank of that matrix is at most r for every $(x,y) \in D$. **Q.E.D.**

We now notice that any polynomial f_i can be written in the form

$$f_i(x,y) = \sum_{(\alpha_1, \ldots, \alpha_n) \in \mathcal{A}} f_{i\alpha}(x) y_1^{\alpha_1} y_2^{\alpha_2} \cdots y_n^{\alpha_n}, \tag{2.2.8}$$

where each $f_{i\alpha}$ is a suitable polynomial. By differentiating both sides of (2.2.8), setting $y = 0$, and comparing with Eq. (2.2.4), we see that for each i, α, there exists a positive constant $c_{i\alpha}$ such that

$$f_{i\alpha}(x) = c_{i\alpha} g_i^{\alpha}(x, 0), \quad \forall x \in \Re^m. \tag{2.2.9}$$

Let us define

$$t = \max_{x \in \Re^n} \text{rank}[\nabla f_{i\alpha}(x) : i = 1, \ldots s; \alpha \in \mathcal{A}] \tag{2.2.10}$$

Using Eq. (2.2.9) we see that

$$\begin{aligned} t &= \max_{x \in \Re^n} \text{rank}[\nabla_x g_i^{\alpha}(x, 0) : i = 1, 2, \ldots, s; \alpha \in \mathcal{A}] \\ &\leq \max_{(x,y) \in \Re^m \times \Re^n} \text{rank}[\nabla_x g_i^{\alpha}(x, y) : i = 1, 2, \ldots, s; \alpha \in \mathcal{A}]. \end{aligned} \tag{2.2.11}$$

Corollary 2.2.1 $C_{rpoly}(\vec{f}; \Re^m \times \Re^n) \geq C_{\infty}(\vec{f}; \Re^m \times \Re^n) \geq t.$

Proof: The first inequality is trivial since we are considering a restricted class of protocols. The second follows from (2.2.6) and (2.2.11). **Q.E.D.**

We make a short digression to verify that the bound t of Corollary 2.2.1 is a generalization of Theorem 3.1.

Theorem 2.2.3 *For the case $s = 1$, that is, for the problem of computing a single polynomial $f(x, y) = \sum_{\alpha \in \mathcal{A}} f_{\alpha}(x) y_1^{\alpha_1} \cdots y_n^{\alpha_n}$, the value of t is equal to the right-hand side of Eq. (2.2.3).*

Proof: Let us fix some $x^* \in \Re^m$. Let $r(x^*)$ be the dimension of the span of $\{\frac{\partial f}{\partial x_j}(x^*, y), j = 1, \ldots, m\}$, where the span is formed in the vector space of functions of the variable y. We only need to show that $\max_{x^* \in \Re^n} r(x^*) = t$. Notice that

$$\nabla_x f(x^*, y) = \sum_{\alpha \in \mathcal{A}} \nabla_x f_{\alpha}(x^*) y_1^{\alpha_1} y_2^{\alpha_2} \cdots y_n^{\alpha_n}.$$

Using the definition of $r(x^*)$, we see that there exist $m - r(x^*)$ linearly independent vectors $\mu_1, \mu_2, \ldots, \mu_{m-r(x^*)}$ in $\Re^m$ that are orthogonal to $\nabla_x f(x^*, y)$ for all y. This is clearly equivalent to

$$\mu_i^T \nabla_x f_{\alpha}(x^*) = 0, \quad \forall \alpha, \ i,$$

and implies that $\text{rank}[\nabla_x f_\alpha : \alpha \in \mathcal{A}] \leq r(x^*)$. Taking the maximum over all x^*, we have $t \leq r(x^*)$. The proof of the reverse inequality is just the reverse of the preceding argument. **Q.E.D.**

We now come to the main result of this section which shows that the lower bound of Corollary 2.2.1 is quite tight.

Theorem 2.2.4 *There exists an open set $D_0 \subset \Re^m$ whose complement has Lebesgue measure zero and such that $C_{rpoly}(\vec{f}; D_0 \times \Re^n) \leq t + 1$.*

Proof: We will show the existence of an open set D_0 and of a set of polynomial message functions $m_1, m_2, \ldots, m_{t+1}$, such that each $f_{i\alpha}$ can be expressed in the form

$$f_{i\alpha}(x) = h_{i\alpha}(m_1(x), \ldots, m_{t+1}(x)), \quad \forall x \in D_0, \tag{2.2.12}$$

where $h_{i\alpha}$ is a suitable rational function. In light of Eq. (2.2.8), processor P_2 is able, upon receipt of the messages $m_1(x), m_2(x), \ldots, m_{t+1}(x)$, to evaluate $f_i(x, y)$ for each i, and this will prove that $C_{rpoly}(\vec{f}; D_0 \times \Re^n) \leq t + 1$, as desired.

Let $F_1 = \Re$ (the field of real numbers). Let $F_3 = F_1(\{f_{i\alpha}\})$ be the field generated by the polynomials $\{f_{i\alpha} : i = 1, \ldots, s; \alpha \in \mathcal{A}\}$ over F_1. Since F_1 has characteristic 0 and F_3/F_1 is finitely generated, Theorem 2.1.5 applies and shows that

$$\text{tr.d.}F_3/F_1 = \dim \mathcal{D}_{F_3/F_1}. \tag{2.2.13}$$

Notice that we are dealing with the situation considered in the example of Section 2. In particular, Eq. (2.1.4) shows that

$$\dim \mathcal{D}_{F_3/F_1} = \max_{x \in \Re^m} \text{rank}[\nabla f_{i\alpha}(x) : i = 1, \ldots, s; \alpha \in \mathcal{A}]. \tag{2.2.14}$$

By comparing with Eq. (2.2.10), we see that $t = \dim \mathcal{D}_{F_3/F_1}$ and using Eq. (2.2.13), we obtain

$$t = \text{tr.d.}F_3/F_1.$$

Let us choose a set of indices such that

$$t = \max_{x \in \Re^m} \text{rank}[\nabla f_{i_1\alpha_1}(x), \ldots, \nabla f_{i_t\alpha_t}(x)],$$

and let F_2 stand for the field generated by $f_{i_1\alpha_1}, \ldots, f_{i_t\alpha_t}$ over F_1. By repeating the argument in the preceding paragraph, we obtain $t = \dim \mathcal{D}_{F_2/F_1} = \text{tr.d.}F_2/F_1$. We then invoke Theorem 2.1.3 to obtain

$$t = \text{tr.d.}F_3/F_1 = \text{tr.d.}F_2/F_1 + \text{tr.d.}F_3/F_2 = t + \text{tr.d.}F_3/F_2,$$

which shows that $\text{tr.d.}F_3/F_2 = 0$.

We notice that F_3 is a finitely generated extension of F_2, and F_2 clearly has characteristic zero. Therefore, we are in a position to apply Theorem 2.1.5 to F_3/F_2, to conclude that F_3/F_2 is a separable algebraic field extension. By Theorem 2.1.1, F_3/F_2 is also a finite algebraic extension. We can therefore apply the theorem of primitive element (Theorem 2.1.2) to F_3/F_2. This leads to the conclusion that $F_3 = F_2(f^*)$ where f^* is some linear combination (over the field F_2) of the polynomials $\{f_{i\alpha} : (i,\alpha) \neq (i_k, \alpha_k),\ \forall k.\}$. More precisely,

$$f^* = \sum_{\alpha \in \mathcal{A}} \sum_{i=1}^{s} \epsilon_{i\alpha} f_{i\alpha}, \tag{2.2.15}$$

where each $\epsilon_{i\alpha}$ is an element of F_2 and where $\epsilon_{i_k\alpha_k} = 0$ for $k = 1, \ldots, t$. In particular, using the definition of F_2, each $\epsilon_{i\alpha}$ can be expressed as a rational function of $f_{i_1\alpha_1}, \ldots, f_{i_t\alpha_t}$.

Since $F_3 = F_2(f^*) = F_1(f_{i_1\alpha_1}, \ldots, f_{i_t\alpha_t}, f^*)$, it follows that each $f_{i\alpha}$ can be expressed as a rational function of the functions $f_{i_1\alpha_1}, \ldots, f_{i_t\alpha_t}, f^*$. Thus, there exist rational functions $\bar{h}_{i\alpha}$ such that

$$f_{i\alpha} = \bar{h}_{i\alpha}(f_{i_1\alpha_1}, \ldots, f_{i_t\alpha_t}, f^*) \tag{2.2.16}$$

Note that (2.2.16) is similar to (2.2.12) except that it refers to the equality of two elements in F_3 and that f^* need not be a polynomial. Let S be the set in $\Re^m$ on which the denominator of some of the rational functions under consideration vanishes. The set S has measure zero. Let us denote the complement of S by D_0. Clearly, D_0 is an open set. By evaluating both sides of Eq. (2.2.16) at an arbitrary vector $x \in D_0$, Eq. (2.2.12) is obtained, provided that we can replace f^* by a polynomial.

To see that f^* can be replaced by a polynomial, we recall the representation (2.2.15) of f^*. Since each $\epsilon_{i\alpha}$ is a rational function of $f_{i_1\alpha_1}, \ldots, f_{i_t\alpha_t}$, the function f^*

can be expressed as the ratio of two polynomials, $f^* = p/q$, where q is a common multiple of the denominators of each one of the rational functions $\epsilon_{i\alpha}$. It follows that q is a polynomial function of $f_{i_1\alpha_1}, \ldots, f_{i_t\alpha_t}$. Let us consider the one–way protocol defined by $m_k = f_{i_k\alpha_k}$, $k = 1, \ldots, t$ and $\bar{m}_{t+1} = f^*$. Then, q is known to a processor who has already received the values of $f_{i_1\alpha_1}, \ldots, f_{i_t\alpha_t}$. Consequently, transmitting the value $p(x)$ (as the last message) carries the same information as transmitting the value $f^*(x)$. We have therefore constructed a one–way protocol (with $m_k = f_{i_k\alpha_k}$, $k = 1, \ldots, t$, and $m_{t+1} = p$) which uses $t+1$ messages, and all messages are polynomial functions of the input x. Furthermore, by Eq. (2.2.16) and the fact that q is a polynomial function of $m_1, \ldots, m_k$, we see that Eq. (2.2.12) holds for some suitable rational functions $h_{i\alpha}$. **Q.E.D.**

In order to turn Theorem 2.2.4 into a useful result, one needs a computationally effective method for evaluating f^* and for constructing a protocol that uses $t+1$ messages. The solution to this problem is not apparent and depends on the structure of the field F_3. However, our proof does suggest a randomized procedure, which we now outline. Assuming that the number of functions $f_{i\alpha}$ is not excessive, we can evaluate the rank of the matrix consisting of the gradients $\nabla_x f_{i\alpha}$ at a random point. Obviously, except for a closed set of zero measure (an algebraic set) we will find the maximum rank t, as well as polynomials $f_{i_1\alpha_1}, \ldots, f_{i_t\alpha_t}$ with the desired properties. Moreover, according to the remark following Theorem 2.1.2, we know that the overwhelming majority of choices of the coefficients $\epsilon_{i\alpha}$ in Eq. (2.2.15) are acceptable.

To illustrate that the additional message f^* is needed, we consider the following example:

Example: Consider the problem of computing $f(x,y) = f_1(x)y_1 + f_2(x)y_2 + \cdots + f_n(x)y_n$, where $x, y \in \Re^n$ and

$$\begin{aligned} f_1 &= x_1^2 x_2 (x_1 + \cdots + x_n), \\ f_2 &= x_1 x_2^2 (x_1 + \cdots + x_n)^2, \end{aligned}$$

and $f_i = (x_1 x_2 (x_1 + \cdots + x_n))^{i+1}$, $i = 3, \ldots, n$. With a little calculation, we see that $t = 2$ and that $\max_{x \in \Re^n} \text{rank}[\nabla f_1(x), \nabla f_2(x)] = 2$. Following the same notation as before, let $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ be, respectively, the field of real numbers, the field gener-

ated by the polynomials f_1, f_2 over $\Re$ and the field generated by the polynomials $f_1, \ldots, f_n$ over $\Re$. Since $\mathcal{F}_2$ contains the element $f_1 f_2 = (x_1 x_2 (x_1 + \ldots + x_n))^3$ and $f_i = (x_1 x_2 (x_1 + \cdots + x_n))^{i+1}$, $i = 3, \ldots, n$, we see that each f_i $(i = 3, \ldots, n)$ is algebraic over $\mathcal{F}_2$ (see Definition 2.1.1). Thus, $\mathcal{F}_3$ is an algebraic extension of $\mathcal{F}_2$. In fact, we can see that $\mathcal{F}_3 = \mathcal{F}_2(f^*)$, where $f^* = x_1 x_2 (x_1 + \ldots + x_n)$. Thus, it is sufficient to let f_1, f_2 and f^* be the message functions since all the polynomials $f_1, \ldots, f_n$ can be computed on the basis of these three messages. (Note that if we do not require that the final evaluation functions to be rational functions, then two messages, namely f_1 and f_2, are sufficient for computing $f_1, \ldots, f_n$.)

To summarize the results in this subsection, we have shown that (as long as we are willing to disregard a set of points of measure zero) the restriction to polynomial messages can increase the communication complexity by at most one. This is in contrast to the earlier results (Theorem 2.2.1) that asserted the existence of protocols which are not necessarily polynomials and whose domain is only some (possibly very small) open set.

2.2.2 Computing Polynomials of the Form $f(x+y)$

In this section we consider the special case where all of the polynomials $f_i : \Re^n \times \Re^n \mapsto \Re$ to be computed are of the form

$$f_i(x, y) = \hat{f}_i(x + y), \quad i = 1, 2, \ldots, s,$$

where each $\hat{f}_i : \Re^n \mapsto \Re$ is a polynomial. We exploit this special structure and show that linear protocols (i.e., the messages are linear functions of the input) are optimal within the class of protocols that use infinitely differentiable message functions.

Let, as in the preceding section,

$$g_i^{\alpha}(x, y) = \frac{\partial^{\alpha} f_i}{\partial y_1^{\alpha_1} \cdots \partial y_n^{\alpha_n}}(x, y).$$

We view $\hat{f}_i$ as a function of a variable $z \in \Re^n$ and we define

$$\hat{g}_i^{\alpha}(z) = \frac{\partial^{\alpha} \hat{f}_i}{\partial z_1^{\alpha_1} \cdots \partial z_n^{\alpha_n}}(z).$$

Let

$$t = \max_{z \in \Re^n} \text{rank}[\nabla_z \hat{g}_i^\alpha(z) : i = 1, \ldots, s; \alpha \in \mathcal{A}]. \tag{2.2.17}$$

Theorem 2.2.5 $C_\infty(\vec{f}; \Re^n \times \Re^n) = C_{linear}(\vec{f}; \Re^n \times \Re^n) = t$.

Proof: We first prove a lower bound. Using Theorem 2.2.2(b), we have

$$C_\infty(\vec{f}; \Re^n \times \Re^n) \geq \max_{(x,y) \in \Re^n \times \Re^n} \text{rank}[\nabla_x g_i^\alpha(x,y); i, \alpha].$$

We notice that $\hat{g}_i^\alpha(z) = g_i^\alpha(x,y)$ and $\nabla_z \hat{g}_i^\alpha(z) = \nabla_x g_i^\alpha(x,y)$, where $z = x + y$. We thus obtain

$$\begin{aligned} C_\infty(\vec{f}; \Re^n \times \Re^n) &\geq \max_{(x,y) \in \Re^n \times \Re^n} \text{rank}[\nabla_x \hat{g}_i^\alpha(x+y); i, \alpha] \\ &= \max_{z \in \Re^n} \text{rank}[\nabla_z \hat{g}_i^\alpha(z); i, \alpha] \\ &= t, \end{aligned}$$

which proves the lower bound. Given that $C_\infty(\vec{f}; \Re^n \times \Re^n) \leq C_{linear}(\vec{f}; \Re^n \times \Re^n)$, the proof of the theorem will be completed once we establish that $C_{linear}(\vec{f}; \Re^n \times \Re^n) \leq t$.

We first consider the case where $t = n$. In this case, we can use the protocol defined by $m_k(x) = x_k$, $k = 1, \ldots, n$. (That is, processor P_1 transmits its entire vector to processor P_2.) This is clearly a linear protocol with t messages and establishes the desired result for the case $t = n$. Notice also that the case $t > n$ cannot occur since t is the rank of a matrix with n rows.

The proof of the upper bound for the general case ($t \leq n$) proceeds by induction on n. For the basis of the induction, we consider the case where $n = 1$. If $t = n = 1$, then the result is true, by the argument of the preceding paragraph. If on the other hand $t = 0$, then $\nabla_z \hat{h}_i^\alpha(z) = 0$ for all $z \in \Re$ and all i, α. By letting $\alpha = (0, 0, \ldots, 0)$, we see that $\nabla_z \hat{f}_i(z) = 0$ for all z and i. Therefore, each $\hat{f}_i$ is a constant function. In this case, processor P_2 can compute $f_i(x,y)$ for each i, without receiving any messages, and $C_{linear}(\vec{f}; \Re^n \times \Re^n) = 0 = t$, as desired.

We now assume that the result has been proved for $n - 1$ ($n \geq 2$) and we prove it for n as well. The case $t = n$ has already been dealt with and we assume that $t < n$.

Lemma 2.2.1 *If $t < n$, then there exists a nonzero vector $c = (c_1, c_2, \ldots, c_n) \in \Re^n$ such that*

$$\sum_{j=1}^{n} c_j \frac{\partial \hat{f}_i}{\partial z_j}(z) = 0, \quad \forall i, z. \tag{2.2.18}$$

Proof: The left hand side of Eq. (2.2.18) is a polynomial, therefore, it suffices to show that the coefficient corresponding to each term $z_1^{\alpha_1} z_2^{\alpha_2} \cdots z_n^{\alpha_n}$ is identically zero. Let us denote the coefficient corresponding to the term $z_1^{\alpha_1} z_2^{\alpha_2} \cdots z_n^{\alpha_n}$ of $\partial \hat{f}_i / \partial z_j$ by $d_\alpha(ij)$. Then Eq. (2.2.18) becomes equivalent to $\sum_{j=1}^{n} c_j d_\alpha(ij) = 0$ for all i and α.

Let $H(z) = [\nabla_z \hat{h}_i^\alpha(z); i = 1, \ldots, s; \alpha \in \mathcal{A}]$, and consider the matrix $H(0)$. Note that the column of $H(0)$ corresponding to indices i, α, is equal to

$$\alpha! \left(d_\alpha(i1), d_\alpha(i2), \ldots, d_\alpha(in)\right),$$

where $\alpha! \stackrel{\text{def}}{=} \alpha_1! \alpha_2! \cdots \alpha_n!$. (This is because the terms corresponding to $\alpha' \neq \alpha$ are either washed out by the differentiations or are set to zero when we let $z = (0, 0, \ldots, 0)$.) We have $\text{rank} H(0) \leq \max_{z \in \Re^n} \text{rank} H(z) = t < n$. Therefore, there exists a nonzero vector $c = (c_1, \ldots, c_n) \in \Re^n$ which is orthogonal to each one of the columns of $H(0)$. This implies that $\sum_{j=1}^{n} c_j d_\alpha(ij) = 0$ and concludes the proof of the lemma. **Q.E.D.**

Without loss of generality, we assume that $c_n \neq 0$, where c_n is the last coordinate of the nonzero vector c given by Lemma 2.2.1. We define an invertible linear transformation $T : \Re^n \mapsto \Re^n$ by means of the formula

$$Tz = (z_1 + c_1 z_n, z_2 + c_2 z_n, \ldots, z_{n-1} + c_{n-1} z_n, c_n z_n).$$

We will show that this coordinate transformation leads to polynomials that are independent of the last coordinate of their argument, which will then allow us to use the induction hypothesis.

Consider the polynomials $\hat{f}_1', \ldots, \hat{f}_s'$ and $f_1', \ldots, f_s'$ defined by

$$\hat{f}_i'(z) = \hat{f}_i(Tz) = \hat{f}_i(z_1 + c_1 z_n, \ldots, z_{n-1} + c_{n-1} z_n, c_n z_n), \tag{2.2.19}$$

$$f_i'(x, y) = \hat{f}_i'(x + y). \tag{2.2.20}$$

Using the chain rule and Eq. (2.2.18), we see that

$$\frac{\partial \hat{f}_i'}{\partial z_n} = \sum_{j=1}^{n} c_j \frac{\partial \hat{f}_i}{\partial z_j} \equiv 0.$$

Therefore, the polynomials $\hat{f}_i'$ are independent of the last coordinate of their argument and can be viewed as mappings defined on $\Re^{n-1}$ (instead of $\Re^n$).

Given that T is an invertible linear transformation, it is easily seen that the rank of the matrix considered in (2.2.17) does not change if each $\hat{f}_i$ is replaced by $\hat{f}_i'$. We now apply the induction hypothesis to the functions $\vec{f}' = \{f_1', \ldots, f_s'\}$ to conclude that

$$C_{linear}(\vec{f}'; \Re^n \times \Re^n) \leq t.$$

Let the linear functions $m_1'(x), \ldots, m_t'(x)$ correspond to a linear protocol for the problem of evaluating the functions in $\vec{f}'$. It follows that there exist polynomials $g_1', \ldots, g_s'$ such that

$$f_i'(x,y) = g_i'(m_1'(x), \ldots, m_t'(x), y), \quad \forall i, x, y.$$

Therefore,

$$\begin{aligned} f_i(x,y) &= \hat{f}_i(x+y) = \hat{f}_i'(T^{-1}(x+y)) = f_i'(T^{-1}x, T^{-1}y) \\ &= g_i'(m_1'(T^{-1}x), \ldots, m_t'(T^{-1}x), T^{-1}y), \quad \forall i, x, y, \end{aligned}$$

where we have use of the definitions (2.2.19) and (2.2.20). Thus, the functions $m_1, \ldots, m_t$ defined by $m_i(x) = m_i'(T^{-1}x)$, $i = 1, \ldots t$, define an one-way protocol for the problem of evaluating $f_1, f_2, \ldots, f_s$. Furthermore, each m_i is linear, since it is the composition of linear functions. Therefore, $C_{linear}(\vec{f}; \Re^n \times \Re^n) \leq t$. This completes the induction and the proof of the theorem. **Q.E.D.**

We remark that the proof of Theorem 2.2.5 actually provides a procedure for constructing a linear and optimal protocol. Furthermore, the proof shows that we do not need to evaluate $\max_{z \in \Re^n} \text{rank} H(z)$ but only the rank of $H(0)$. If the latter rank is equal to n, the problem is trivial, and if it is less than n, Lemma 3.1 applies and the problem can be reduced to one with a smaller dimension. Another point worth mentioning is that our proof actually suggests a deterministic procedure for

constructing the optimal linear protocol. In fact, one can first compute the rank of $H(0)$. If rank$H(0) = n$, then $m_k(x) = x_k$ is an optimal protocol. If $H(0)$ has rank less than n, then one can use, for example, Gaussian elimination method to find a nonzero vector c such that $c^T H(0) = 0$. As shown in the proof, the problem is reduced to one with a smaller dimension by a suitable change of variables. By repeating this process at most finitely many times, one will find an optimal linear protocol for computing functions f_i, $i = 1, \ldots, s$.

2.3 Discussion and Possible Extensions

In this chapter, we have considered the problem of computing a set of polynomials with one–way protocols in which messages are rational/polynomial functions. An almost optimal upper bound is derived by using field extension theory. Our result extends Abelson's result in several directions: (i) Abelson's result applies to computing a single function only, whereas our bound is valid for computing several functions; (ii) The optimal one–way protocol given by Abelson's result is a local one because it is based on the implicit function theorem. In contrast, our result shows that by sending at most one additional message we can construct a protocol whose messages are rational functions and thus the protocol works for all inputs (x, y) except for the set of points at which the denominator of one of the message function vanishes. Clearly, such a set of points has Lebesque measure zero. We then apply our result to the case where the polynomials to be computed are of the form $\hat{f}_j(x + y)$, $j = 1, \ldots, s$, with each $\hat{f}_j$ being some polynomial. We have shown that there exists an optimal one–way protocol for computing $\hat{f}_j(x+y)$, $j = 1, \ldots, s$, in which the messages are linear functions of x.

There are still several unsettled questions concerning the one–way communication complexity for computing polynomials. First, we notice that although all the messages of the protocol we constructed are polynomials, each final evaluation function h_j is in general a rational function. Thus, the protocol is invalid for the points at which one of the denominators of these rational functions vanishes. In some applications, we may be interested in a truly global protocol (for computing polynomials) whose domain is the entire Euclidean space $\Re^m$. For example, we may

wish to require that all the message functions as well as final evaluation functions be polynomials. We have seen that we will increase the communication complexity by at most 1 if we allow the possibility of each final evaluation function h_j being a rational function. It is therefore interesting to find out the minimum number of messages needed for computing polynomials with a portocol in which both the messages and the final evaluation functions are polynomilas. In particular, we wish to see if there exists some constant C, such that

$$C_{ppoly}(\vec{f}; \Re^m \times \Re^n) \leq t + C,$$

where t is given by Eq. (2.2.10). We believe that the answer to this question is negative. Another unsettled question related to our result has to do with the actual construction of the protocol given in Theorem 2.2.4. As mentioned before, there is a randomized algorithm for constructing this protocol. It is of interest to study for any given set of polynomials whose coefficients are rational numbers, if there exists a deterministic polynomial time algorithm (in terms of the size of the coefficients of the polynomials) for constructing a protocol that uses at most $t + 1$ messages. Finally, it remains to be seen if there exist any interesting applications of our result in areas such as digital signal processing, system theory and etc.

One may be able to improve our result in another direction, namely, by considering the problem of computing more general classes of functions such as rational functions. One can, of course, always compute a rational function, say, $f = p/q$, by computing its numerator p and the denominator q separately and applying the results presented in this chapter. But, this may be wasteful since there may exist a protocol which uses fewer messages by computing the polynomials $p \cdot g$ and $q \cdot g$, where g is some nonzero polynomial. (Clearly, processor P_2 can recover f from the polynomials $p \cdot g$ and $q \cdot g$.) At present, we are not aware of any algebraic technique for studying this problem.

Chapter 3

Two–Way Continuous Communication Model

In this chapter, we study the two–way communication complexity of computing a function $f(x,y)$ using continuous communicaton protocols as defined in Section 1.1.1. In particular, we assume that f has some algebraic structure, for example, f may be rational fucntion, and accordingly we consider the protocols with similar algebraic properties, namely, purely rational protocols, polynomial protocols and so on. We establish several new lower bounds that improve upon a result by Abelson ([A 80]). Our results are stronger than Abelson's lower bound since they exploit the algebraic structures present in the problems under consideration. We also provide a new lower bound which makes use of the first order derivatives of f to be computed, unlike Abelson's result which uses only the second order derivatives of f. We then apply our results to the problem of computing a particular entry of $[x+y]^{-1}$, where x and y are some $n \times n$ matrices, and obtain an $\Omega(n^2)$ lower bound, in contrast to the $\Omega(n)$ lower bound obtained by applying Abelson's result. In another application, we obtain a lower bound of n for the problem of computing a root of a polynomial equation of degree $n-1$, under the assumption that each processor only knows part of each coefficient of the polynomial. We also show that Abelson's result can only provide an $\Omega(1)$ lower bound when applied to this problem. The organization is as follows. Section 3.1 provides a brief introduction to some algebraic results that will be needed later, namely, Hilbert's Nullstellensatz and some elements of dimension theory. Section 3.2 contains the proofs of several new lower bounds. In addition, it also considers an important application where

the problem is to compute some particular entry of the inverse matrix $[x+y]^{-1}$. Finally, in Section 3.5 we give a new lower bound approach for determining the two–way communication complexity. We also use the new lower bound to examine the communication complexity of solving a polynomial equation and prove a tight lower for a broad class of communication protocols. We also compare this new lower bound against that of Abelson's.

3.1 Algebraic Preliminaries

In this section, we give some algebraic background results (e.g. Hilbert's Nullstellensatz) from algebraic geometry. (see e.g. [AM 69,H 77]) that will be needed in Section 3.2.

Let $C[x_1, x_2, \ldots, x_n]$ denote the ring of polynomials of variables $x_1, \ldots, x_n$ over C, the field of complex numbers. Let $f, g \in C[x_1, x_2, \ldots, x_n]$. We use the notation $f|g$ and say that f divides g if there exists some $h \in C[x_1, x_2, \ldots, x_n]$ such that $g = f \cdot h$. We say that a polynomial $g \in C[x_1, x_2, \ldots, x_n]$ is irreducible if $g = f \cdot h$ implies that either f or h is an element of C. As is well known, $C[x_1, x_2, \ldots, x_n]$ is a unique factorization ring, that is, each one of its elements can be expressed as a product of irreducible polynomials. Furthermore, this factorization is unique up to reordering of the factors and up to multiplication of each factor by an element of C.

Let $f_1, \ldots, f_r$ be some polynomials in $C[x_1, x_2, \ldots, x_n]$. We define the zero set of $f_1, \ldots, f_r$ by

$$V(f_1, \ldots, f_r) = \{(x_1, x_2 \ldots, x_n) \in C^n |\ f_k(x_1, x_2 \ldots, x_n) = 0,\ 1 \leq k \leq r\}.$$

We now state a simple version of Hilbert's Nullstellensatz [AM 69, page 85] that will be used in Section 3.2.

Theorem 3.1.1 (Hilbert's Nullstellensatz) *Let $f_1, \ldots, f_r$ be some polynomials in $C[x_1, \ldots, x_n]$ If $g \in C[x_1, \ldots, x_n]$ and $V(f_1, \ldots, f_r) \subset V(g)$, then there exist some polynomials $g_1, \ldots, g_r \in C[x_1, \ldots, x_n]$ and some positive integer k such that*

$$g^k = g_1 f_1 + g_2 f_2 + \cdots + g_r f_r. \tag{3.1.1}$$

Notice that if Eq. (3.1.1) holds, then $V(f_1,\ldots,f_r) \subset V(g^k) = V(g)$. The fact that the converse is also true is exactly the content of Hilbert's theorem.

Corollary 3.1.1 *If $f, g \in C[x_1,\ldots,x_n]$, and if $f(x) = 0$ implies that $g(x) = 0$, i.e., if $V(f) \subset V(g)$, then there is an integer k and some $h \in C[x_1,\ldots,x_n]$ such that $g^k = fh$. (In other words, $f|g^k$.)*

One can assign a topology to the field C^n by taking the family of sets

$$\{V(S) \mid S \text{ is an ideal}^1 \text{ of } C[x_1,\ldots,x_n]\}$$

as the closed sets. (It is a simple exercise to check that these sets satisfy the usual requirements for the closed sets of a topology.) Traditionally, this topology is called the Zariski topology on C^n. It can be seen easily that C^n is an irreducible closed set under the Zariski topology. the Zariski closed sets are also called *algebraic sets.* An important property of Zariski topology is the following (see [H 77]).

Theorem 3.1.2 *Every two nonempty Zariski open sets of C^n have nonempty intersection and every closed set has zero Lebesgue measure. Moreover, if S is an irreducible closed set of C^n, then every two nonempty open subsets of S (in relative Zariski topology) have nonempty intersection.*

3.2 Two–Way Communication Complexity

In this section we study the two–way communication complexity of evaluating a function $f : D_f \mapsto C$, where D_f, the domain of f is an open subset of $C^m \times C^n$. (In Subsection 3.2.2, we also consider the problem of computing a real–valued function f defined on $\Re^m \times \Re^n$.) Throughout, we assume that f is twice continuously differentiable on D_f.

[1] S is called an ideal of $C[x_1,\ldots,x_n]$ iff (i) S is closed under addition; (ii) $gS \subseteq S$, for all $g \in C[x_1,\ldots,x_n]$.

3.2.1 Abelson's Lower Bound

We first fix some notations.

Definition 3.2.1 *We let $H_{xy}(f)$ be the matrix (of size $m \times n$) whose (i,j)-th entry is given by $\frac{\partial^2 f}{\partial x_i \partial y_j}$. We use the alternative notations $(H_{xy}(f))(p)$ and $H_{xy}(f)|_p$ to denote the value of $H_{xy}(f)$ at some vector $p \in D_f$. Also, we let $\nabla_x f$ and $\nabla_y f$ stand for the vectors of dimensions m and n (respectively) with the partial derivatives of f with respect to the components of x and y, respectively.*

The following basic result has been established by Abelson [A 80]:

Theorem 3.2.1 *For any open set $D \subset D_f$ and any $p \in D$, we have*

$$C_2(f;D) \geq \text{rank}(H_{xy}(f))(p). \tag{3.2.1}$$

Remark: This result was actually proved in [A 80] for real–valued functions defined on $\Re^{m+n}$ but the proof remains valid when $\Re$ is replaced by C. We will use, in Subsection 3.2.2, the original version of Theorem 3.2.1 in which all functions are real–valued.

Theorem 3.2.1 has an obvious corollary:

Corollary 3.2.1 *For any open set $D \subset D_f$, we have*

$$C_2(f;D) \geq \max_{p \in D} \text{rank}(H_{xy}(f))(p). \tag{3.2.2}$$

The matrix $H_{xy}(f)$ is defined in terms of the cross derivatives of f and in some sense provides information on how x and y are interrelated in the formula for $f(x,y)$. On the other hand, Eq. (3.2.1) only takes into account the second order derivatives of f and ignores the higher order derivatives or the first order derivatives of f. Thus, this bound should not be expected to be tight, in general. As an example, let f be a linear function, e.g, $f(x,y) = a^T x + b^T y$ $(a \in C^m, b \in C^n, a \neq 0, b \neq 0)$. It is clear that $C_2(f;D) = 1$, for any open set D, while Eq. (3.2.1) gives a vacuous lower bound of zero. The following corollary strengthens Eq. (3.2.1) somewhat, by incorporating the first order derivatives of f as well. It is only a minor improvement because it can increase the lower bound by at most 1.

Corollary 3.2.2 *For any open set $D \subset D_f$ we have*

$$C_2(f;D) \geq \max_{c \in C} \max_{p \in D} \text{rank}[(H_{xy}(f))(p) + c\nabla_x f(p) \cdot \nabla_y f(p)^T].$$

Proof: We notice that $C_2(f;D) \geq C_2(g \circ f;D)$, for any twice continuously differentiable function $g : C \mapsto C$, where $g \circ f$ denotes the composition of f and g. For any $p \in D$, and $c \in C$, consider a function g such that $g'(f(p)) \neq 0$ and $c = g''(f(p))/g'(f(p))$. The result then follows by applying Theorem 3.2.1 to the function $g \circ f$. **Q.E.D.**

In the remainder of this section, we investigate the extent to which Abelson's bound is tight and we derive some tighter bounds. We will mostly restrict attention to the case where f is a rational function and we will require the messages to be rational functions of the input. In the next subsection, we identify two instances where Abelson's lower bound (Theorem 3.2.1) is tight. Then, in Subsection 3.2.3, we establish some new general lower bounds by making use of Hilbert's Nullstellensatz.

3.2.2 Some Cases Where Abelson's Bound is Tight

We consider here two particular cases in which Abelson's bound (Theorem 3.2.1) can be shown to be tight. This is in contrast to the results of Section 3.5 in which it will be shown to be far from tight.

Theorem 3.2.2 *Suppose that $f(x,y) = x^T Q y$, where Q is a matrix of size $m \times n$ and $x \in \Re^m$, $y \in \Re^n$. Then $C_2(f;\Re^{n+m}) = \text{rank}H_{xy}(f) = \text{rank}(Q)$. In fact, the lower bound can be attained by a one–way protocol with linear messages.*

Proof: Let $\text{rank}(Q) = r$. By Theorem 3.2.1, we see that $C_2(f;\Re^{n+m}) \geq \text{rank}H_{xy}(f) = \text{rank}(Q) = r$. To prove the other direction of the inequality, we will present a one–way linear protocol that uses exactly r messages. Using the singular value decomposition of Q, there exist vectors $u_1,\ldots,u_r \in \Re^m$ and $v_1,\ldots,v_r \in \Re^n$ such that

$$Q = u_1 v_1^T + u_2 v_2^T + \cdots + u_r v_r^T,$$

from which we obtain

$$x^T Q y = x^T u_1 v_1^T y + x^T u_2 v_2^T y + \cdots + x^T u_r v_r^T y. \tag{3.2.3}$$

Notice that in Eq. (3.2.3) each one of the expressions $x^T u_i$ and $v_i^T y$ is a scalar. Thus, the one-way protocol with r linear messages, defined by $m_i(x) = x^T u_i$, $i = 1, \ldots, r$, is adequate for computing f. **Q.E.D.**

Theorem 3.2.2 states that Abelson's bound is tight for homogeneous[2] quadratic polynomials. What happens for homogeneous polynomials of degree greater than 2? In what follows, we will show the tightness of Abelson's bound for computing functions of the form: $f(x, y) = g(x + y)$, where g is a nonlinear homogeneous polynomial in no more than 4 variables. While this result determines a case for which $C_2(f; \Re^{2n})$ can be determined completely, it is of little use in practice. This is because we have $n \leq 4$ and the naive protocol $m_i(x) = x_i$, $i = 1, \ldots, n$, uses at most 4 messages and cannot be too far from being optimal. Our result makes use of the following theorem proved by Gordan and Nöether in 1876 ([GN 76]).

Theorem 3.2.3 *Let $f : \Re^n \mapsto \Re$ be a nonlinear homogeneous polynomial in $n \leq 4$ variables and let $H(f)$ be its Hessian matrix. If $\det H(f) \equiv 0$, then there exists a linear mapping T from $\Re^n$ onto $\Re^{n-1}$ and a homogeneous polynomial $g : \Re^{n-1} \mapsto \Re$ such that $f(x) = g(Tx)$.*

Our result is the following:

Theorem 3.2.4 *Let $g : \Re^n \mapsto \Re$ be a nonlinear homogeneous polynomial and let the polynomial $f : \Re^{2n} \mapsto \Re^n$ be defined by $f(x, y) = g(x + y)$. If $n \leq 4$, then*

$$C_2(f; \Re^{2n}) = \max_{(x,y) \in \Re^{2n}} \text{rank} H_{xy}(f)|_{(x,y)} = C_{linear}(f; \Re^{2n}).$$

Proof: Let $z = x + y$. We regard g as a nonlinear polynomial in the variable $z \in \Re^n$. Let k be the smallest integer such that there exists some linear mapping T from $\Re^n$ onto $\Re^k$ and some homogeneous polynomial $\hat{g} : \Re^k \mapsto \Re$ such that $g(z) = \hat{g}(Tz)$.

[2] A polynomial $f(x)$ is *homogeneous* of degree k if $f(tx) = t^k f(x)$, for all $t \in \Re$ and $x \in \Re^n$.

Since g is nonlinear and T is linear, we see that $\hat{g}$ is also nonlinear. We claim that there exists some vector $\hat{z} = (\hat{z}_1, \ldots, \hat{z}_k) \in \Re^k$ at which $H(\hat{g})$ is nonsingular. Indeed, if this is not so, then by Theorem 3.2.3, there exists another linear mapping $\bar{T}$ from $\Re^k$ onto $\Re^{k-1}$ and some homogeneous polynomial $\bar{g} : \Re^{k-1} \mapsto \Re$ such that $\hat{g}(\hat{z}) = \bar{g}(\bar{T}\hat{z})$. But this implies that $g(z) = \bar{g}(\bar{T}Tz)$. Since the composition of T and $\bar{T}$ maps $\Re^n$ onto $\Re^{k-1}$, this contradicts the definition of k.

A simple calculation shows that $H(g)|_z = T^T H(\hat{g})|_{Tz} T$. Since T maps $\Re^n$ onto $\Re^k$, the matrices T and T^T have full rank and we obtain $\text{rank}H(g)|_z = \text{rank}H(\hat{g})|_{Tz}$. Since the range of T is all of $\Re^k$, we have

$$\max_{z \in \Re^n} \text{rank}H(g)|_z = \max_{\hat{z} \in \Re^k} \text{rank}H(\hat{g})|_{\hat{z}} = k.$$

Since $H_{xy}(f)|_{(x,y)} = H(g)|_{z=x+y}$, we obtain that $\max_{x,y} \text{rank}H_{xy}(f) = k$. It then follows from Theorem 3.2.1 that $C_2(f; \Re^{2n}) \geq k$. To establish the reverse inequality, we will present a protocol for computing f that uses exactly k messages. Let $m_i(x) = T_i x$, $i = 1, \ldots, k$, where T_i is the ith row of the matrix T. Then,

$$\begin{aligned} f(x,y) &= g(x+y) = \hat{g}(T(x+y)) = \hat{g}(T_1(x+y), \ldots, T_k(x+y)) \\ &= \hat{g}\left(m_1(x) + T_1 y, \ldots, m_k(x) + T_k y\right). \end{aligned}$$

This last formula shows that f can be computed using the one-way protocol with messages $m_i(x) = T_i x$, $i = 1, \ldots, k$. In particular, $C_2(f; \Re^{2n}) \leq C_{linear}(f; \Re^{2n}) \leq k$, which completes the proof. **Q.E.D.**

Unfortunately, Theorem 3.2.4 is not true for the case $n > 4$, for the simple reason that Theorem 3.2.3 fails to hold. Historically, Hess had published a paper in which he gave an erroneous proof of Theorem 3.2.3 for all n. It was later discovered by Gordan and Nöether that Hess' proof was incorrect and proved that the largest value of n for which Theorem 3.2.3 holds is 4. In particular, they gave the following counterexample.

Example: Consider the homogeneous polynomial $f(x_1, x_2, x_3, x_4, x_5) = x_1 x_4^2 + 2x_2 x_4 x_5 + x_3 x_5^2$. It can be easily calculated that

$$H(f) = \begin{bmatrix} 0 & 0 & 0 & 2x_4 & 0 \\ 0 & 0 & 0 & 2x_5 & 2x_4 \\ 0 & 0 & 0 & 0 & 2x_5 \\ 2x_4 & 2x_5 & 0 & 2x_1 & 2x_2 \\ 0 & 2x_4 & 2x_5 & 2x_2 & 2x_3 \end{bmatrix}. \tag{3.2.4}$$

From the above expression, we see that that $\det(H(f)) \equiv 0$. Thus, the maximum rank of $H(f)$ is at most 4. In what follows we show that there does not exist a linear transformation $T : \Re^5 \mapsto \Re^k$ $(k \leq 4)$ such that $f(x) = g(Tx)$, where $x = (x_1, x_2, x_3, x_4, x_5)^T$ and g is some homogeneous polynomial. In fact, if such T exists, then we let $S = \{x \mid Tx = 0\}$. Since $k \leq 4$, we see that S has dimension at least one. Let c be a nonzero vector in S and $t \in \Re$. For each $x \in \Re^5$, let us define a function $\bar{f}_x(t) \triangleq f(x + tc)$. Since $f(x + tc) = g(T(x + tc)) = g(Tx)$, it follows that $\bar{f}_x(t)$ is a constant function of t, for all $x \in \Re^5$. This implies that $\frac{d\bar{f}_x}{dt}|_{t=0} = 0$, $\forall x \in \Re^5$. In other words, we have

$$c_1 \frac{\partial f}{\partial x_1} + c_2 \frac{\partial f}{\partial x_2} + \cdots + c_5 \frac{\partial f}{\partial x_5} \equiv 0, \qquad \forall x \in \Re^5.$$

By differentiating the above relation further, we see that

$$c_1 \frac{\partial^2 f}{\partial x_1 \partial x_i} + c_2 \frac{\partial^2 f}{\partial x_2 \partial x_i} + \cdots + c_5 \frac{\partial^2 f}{\partial x_5 \partial x_i} \equiv 0, \qquad \forall x \in \Re^5,\ i = 1, \ldots, 5.$$

Thus, $H(f)c \equiv 0$. Such condition can be easily shown to imply that $c = 0$, by using the expression (3.2.4). Contradiction! **Q.E.D.**

3.2.3 General Properties

In this subsection we present some general properties of the two–way communication protocols which will be needed in future. Suppose that $\vec{f} = \{f_1(x, y), \ldots, f_s(x, y)\}$ are some rational functions of (x, y). For $j = 1, \ldots, s$, let $f_j(x, y) = p_j(x, y)/q_j(x, y)$, where p_j and q_j are two relative prime polynomials. We define $D_{\vec{f}} = \{(x, y) \in C^{m+n} \mid q_j(x, y) \neq 0,\ j = 1, \ldots, s.\}$. Let D be an open subset of $D_{\vec{f}}$.

Theorem 3.2.5 $C_{prat}(\vec{f}; D) \leq C_{rpoly}(\vec{f}; D) \leq 2C_{prat}(\vec{f}; D)$.

Proof: The first inequality is trivial since we are considering a more restricted class of protocols. For the second inequality, we only need to show that for any rational protocol $\pi \in \Pi_{prat}(\vec{f}; D)$ there exists a polynomial protocol $\pi' \in \Pi_{rpoly}$ such that $r(\pi') \leq 2r$. Let $r = r(\pi)$ and let $m_1, m_2, \ldots, m_r$ be the message functions of π. By assumption, each $m_i(x, y)$ is a rational function. Furthermore, we let $T_{1\to 2}$ (respectively, $T_{2\to 1}$) denote the set of i's for which the ith message is transmitted from P_1 to P_2 (respectively, from P_2 to P_1). Since a message can only be a function of the information available to that processor, we have

$$m_i(x, y) = \hat{m}_i\left(x, m_1(x, y), \ldots, m_{i-1}(x, y)\right), \qquad \text{if } i \in T_{1\to 2},$$

and

$$m_i(x, y) = \hat{m}_i\left(y, m_1(x, y), \ldots, m_{i-1}(x, y)\right), \qquad \text{if } i \in T_{2\to 1}.$$

where each $\hat{m}_i$ is a rational function of $m_1, \ldots, m_{i-1}$ and x (or y, depending on if $i \in T_{1\to 2}$ or $i \in T_{2\to 1}$). Without loss of generality, we assume that processor P_1 does the final computations

$$f_j(x, y) = h_j\left(x, m_1(x, y), \ldots, m_r(x, y)\right), \qquad \forall (x, y) \in D,\ j = 1, \ldots, s, \tag{3.2.6}$$

where each h_j is some rational function. Let us write $\hat{m}_i = \alpha_i/\beta_i$, where α_i and $beta_i$ are some polynomials. Suppose that the highest degree of $\alpha_i, \beta_i, i = 1 \ldots, r$ is k. Let us define as follows a new polynomial protocol π' which has $2r$ messages $m'_1, m'_2, \ldots, m'_{2r}$.

$$m'_1(x, y) = \alpha_1(x, y), \qquad m'_2(x, y) = \beta_1(x, y). \tag{3.2.7}$$

For $i \in T_{1\to 2}$ and $i < r$, we define

$$m'_{2i+1}(x, y) = \left(\prod_{j=1}^{i} m'_{2j}\right)^k \alpha_{i+1}\left(x, \frac{m'_1}{m'_2}, \ldots, \frac{m'_{2i-1}}{m'_{2i}}\right), \tag{3.2.8}$$

and

$$m'_{2i+2}(x, y) = \left(\prod_{j=1}^{i} m'_{2j}\right)^k \beta_{i+1}\left(x, \frac{m'_1}{m'_2}, \ldots, \frac{m'_{2i-1}}{m'_{2i}}\right). \tag{3.2.9}$$

In a similar fashion, we can define the message functions $m'_{2i+1}(x,y)$ and $m'_{2i+2}(x,y)$ for the case $i \in T_{2\to 1}$.

We will show, by induction on i, that $m_i(x,y) = m'_{2i-1}(x,y)/m'_{2i}(x,y)$ and that m'_{2i-1}, m'_{2i} are polynomials. The case when $i = 1$ is obvious from Eq. (3.2.7). Suppose that the inductive hypothesis holds true for $i \leq j$. Without loss of generality, we assume that $j+1 \in T_{1\to 2}$. By Eqs. (3.2.8) and (3.2.9), we see that

$$m'_{2j+1}(x,y) = \left(\prod_{i=1}^{j} m'_{2i}(x,y)\right)^k \alpha_{j+1}\left(x, m_1, \ldots, m_j\right)$$

and

$$m'_{2j+2}(x,y) = \left(\prod_{i=1}^{j} m'_{2i}(x,y)\right)^k \beta_{j+1}\left(x, m_1, \ldots, m_j\right).$$

Thus, it follows from Eq. (3.2.5) that

$$\frac{m'_{2j+1}(x,y)}{m'_{2j+2}(x,y)} = \frac{\alpha_{j+1}\left(x, m_1, \ldots, m_j\right)}{\beta_{j+1}\left(x, m_1, \ldots, m_j\right)} = m_{j+1}.$$

The case when $j+1 \in T_{2\to 1}$ can be dealt similarly.

We now show that m'_{2j+1}, m'_{2j+2} are polynomials. By the inductive hypothesis, each m'_i $(i \leq 2j)$ is a polynomial of x and y. Since the degree of the polynomial α_{j+1} is no more than k, we see that the denominator of the rational function $\alpha_{j+1}\left(x, \frac{m'_1}{m'_2}, \ldots, \frac{m'_{2j-1}}{m'_{2j}}\right)$ can be written as a product of polynomials m'_{2i} $(i \leq j)$ and must be a factor of $\left(\prod_{i=1}^{j} m'_{2i}\right)^k$. Thus, by from Eqs. (3.2.8) we see that m'_{2j+1} (assuming that $j \in T_{1\to 2}$) is a polynomial. Similarly, we can show that m'_{2j+2} is also a polynomial, which completes the induction. Hence Eqs. (3.2.8) and (3.2.9) defines a polynomial protocol. Finally, since the m_i's can be recovered from the messages $m'_1, \ldots, m'_{2r}$ by using the relations $m_i(x,y) = m'_{2i-1}/m'_{2i}$, $\forall i \leq r$, we see that after the messages $m'_1, \ldots, m'_{2r}$ are transmitted, processor P_1 can compute the rational functions $f_1, \ldots, f_s$ according to Eq. (3.2.6). Because each h_j is a rational function and each m_i is also a rational function of the messages of π', we conclude that $\pi' \in \Pi_{rpoly}(\vec{f}; D)$. **Q.E.D.**

Remark: We can modify the proof of Theorem 3.2.5 to show that for each $j = 1, \ldots, s$, there exists a polynomial g_j such that

$$C_{ppoly}(p_j g_j; D) \leq 2C_{prat}(\vec{f}; D) \tag{3.2.10}$$

$$C_{ppoly}(q_j g_j; D) \leq 2C_{prat}(\vec{f}; D). \tag{3.2.11}$$

In fact, let us fix some j and write $h_j = \frac{\alpha_{r+1}}{l}\beta_{r+1}$, where α_{r+1} and β_{r+1} are some polynomials. We can modify the proof of Theorem 3.2.5 as follows: (1) set k to be the maximum degree of $\alpha_i, \beta_i, i = 1, \ldots, r+1$; (2) define the polynomials m'_{2r+1} and m'_{2r+2} using Eq. (3.2.8) and (3.2.9) with $i = r$. Then, by using the same induction argument, we can conclude that m'_{2r+1} and m'_{2r+2} are polynomials and

$$\frac{m'_{2r+1}(x,y)}{m'_{2r+2}(x,y)} = \frac{\alpha_{r+1}(x, m_1, \ldots, m_j)}{\beta_{r+1}(x, m_1, \ldots, m_j)} = f_j = \frac{p_j}{q_j}.$$

Using the unique factorization property of polynomial functions (cf. Section 3.1) and the fact that p_j and q_j are two relatively prime polynomials, we see that there exists a nonzero polynomial g_j such that $m'_{2r+1} = p_j g_j$ and $m'_{2r+2} = q_j g_j$. Meanwhile, in light of Eqs. (3.2.8) and (3.2.9) (with $i = r$), we see that the polynomials m'_{2r+1} and m'_{2r+2} can be evaluated through some polynomial function on the basis of the polynomial messages $m'_1, \ldots, m'_{2r}$. Thus, Eqs. (3.2.10) and (3.2.11) have been established.

Theorem 3.2.5 basically says that the class of polynomial protocols is more or less equally powerful as the class of rational protocols. So, if we are interested only in the asymptotics of the communication complexity, we can put our effort in determining $C_{rpoly}(\vec{f}; D)$ instead of $C_{prat}(\vec{f}; D)$. It is worth mentioning that Theorem 3.2.5 remains true if $\mathcal{C}$ is replaced by $\Re$. In other words, if D is an open subset of $\Re^{m+n}$, $x \in \Re^m$, $y \in \Re^n$ and each f_j is a rational function defined on D, then any two–way communication protocol $\pi \in \Pi_{prat}(\vec{f}; D)$ can still be transformed into a polynomial protocol in $\Pi_{rpoly}(\vec{f}; D)$ by at most doubling the number of messages. It should also be mentioned that our proof is also valid for one–way communication protocols. As a matter of fact, we can simply take $T_{1\to 2}$ (or $T_{2\to 1}$) as an empty set in the previous proof.

Next we consider the problem of computing a set of rational functions $f_i(x,y), i = 1, 2, \ldots, s$ with a purely rational protocol (see Subsection 1.1.1 of Chapter 1). Let

us use $\vec{f}$ to denote $\{f_1, \ldots, f_s\}$ and use $D_{\vec{f}}$ to denote the set of vectors in $\Re^{m+n}$ (or C^{m+n}) such that every rational function f_i $(1 \leq i \leq s)$ is well defined. Let D be an open subset of $D_{\vec{f}}$. Our next result says that if constrained to using purely rational protocols, then the problem of computing $f_1, \ldots, f_s$ over D is more or less equally difficult with computing the partial derivatives of $f_i(x,y), i = 1, 2, \ldots, s$, over D. In what follows we will use the short notation $\frac{\partial \vec{f}}{\partial x_1}$ to denote the set of functions $\{\frac{\partial f_i}{\partial x_j}, i = 1, \ldots, s\}$.

Theorem 3.2.6 *For every $1 \leq j \leq m$, there holds*

$$C_{prat}(\frac{\partial \vec{f}}{\partial x_1}, \vec{f}; D) \leq 2C_{prat}(\vec{f}; D).$$

Proof: Let $\pi \in \Pi_{prat}(\vec{f}; D)$ be a two way communication protocol for computing $f_1, \ldots, f_s$ and let $m_1, m_2, \ldots, m_r$ be the rational message functions of π. We consider a protocol π' for computing each f_j and its derivative as follows. For all $1 \leq i \leq r$, we let

$$m'_{2i-1} = m_i, \quad m'_{2i} = \frac{\partial m_i}{\partial x_1}. \tag{3.2.12}$$

We will show that π' is indeed a protocol, i.e., each message $m'_i(x,y)$ is a function of $m'_k(x,y)$, $k < i$. We prove this by induction. Without loss of generality, we assume m_1 is sent by P_1 which implies m_1 only depends on x. Therefore $\frac{\partial m_1}{\partial x_1}$ can also be computed and sent to P_1. This means that m'_1, m'_2 can be generated by P_1. Now suppose that for $k \leq i$ the messages m'_{2k-1} and m'_{2k} are functions of m'_l's $(l < 2k-1)$ and x (or y, depending on if $i \in T_{1\to 2}$ or $T_{2\to 1}$). Without loss of generality, we assume $i+1 \in T_{1\to 2}$. As a result, we know that m_{i+1} is a function of the form $\hat{m}_{i+1}(x, m_1, \ldots, m_i)$, where $\hat{m}_{i+1}$ is a rational function. By Eq.(3.2.12), this implies that m'_{2i+1} can be computed as a function of $m'_1, \ldots, m'_{2i}$. Furthermore, notice that

$$\frac{\partial m_{i+1}}{\partial x_1} = \frac{\partial \hat{m}_{i+1}}{\partial x_1} + \frac{\partial \hat{m}_{i+1}}{\partial m_1}\frac{\partial m_1}{\partial x_1} + \frac{\partial \hat{m}_{i+1}}{\partial m_2}\frac{\partial m_2}{\partial x_1} + \cdots + \frac{\partial \hat{m}_{i+1}}{\partial m_i}\frac{\partial m_i}{\partial x_1},$$

which implies that

$$m'_{2i+2} = \frac{\partial m_{i+1}}{\partial x_1} = \frac{\partial \hat{m}_{i+1}}{\partial m_1}m'_2 + \frac{\partial \hat{m}_{i+1}}{\partial m_2}m'_4 + \cdots + \frac{\partial \hat{m}_{i+1}}{\partial m_j}m'_{2i} + \frac{\partial \hat{m}_{i+1}}{\partial x_1}. \tag{3.2.13}$$

Since $\frac{\partial \hat{m}_{i+1}}{\partial m_k}$ $(k \leq i)$ is a function of $m'_1, m'_3, \ldots, m'_{2i-1}$ and x, we therefore see that m'_{2i+2} is indeed a function of x and $m'_1, \ldots, m'_{2i}$. This completes the induction. Since the message functions m'_i $(1 \leq i \leq 2r)$ and the functions h_j $(1 \leq j \leq s)$ are clearly rational functions, it follows that π' is a purely rational protocol.

We now show that π' is a legitimate protocol for computing $\vec{f}$ and $\frac{\partial \vec{f}}{\partial x_1}$. Since the messages of π' contain the messages of π and since π is legitimate for computing $\vec{f}$, it follows that π' is also legitimate for computing $\vec{f}$. Therefore, what remains to be shown is that π' is legitimate for computing $\frac{\partial \vec{f}}{\partial x_1}$. Without loss of generality, we assume that processor P_1 gets to compute some rational function $f_j(x,y)$ $(1 \leq j \leq s)$:

$$f_j(x,y) = h_j(x, m_1(x,y), \ldots, m_r(x,y)).$$

Note that

$$\frac{\partial f_j}{\partial x_1} = \frac{\partial h_i}{\partial x_1} + \frac{\partial h_j}{\partial m_1}\frac{\partial m_1}{\partial x_1} + \frac{\partial h_j}{\partial m_2}\frac{\partial m_2}{\partial x_1} + \cdots + \frac{\partial h_j}{\partial m_r}\frac{\partial m_r}{\partial x_1}, \qquad \forall (x,y) \in D. \quad (3.2.14)$$

Since at the end of the communication processor P_1 has complete knowledge of $m_i, \frac{\partial m_i}{\partial x_1}, i = 1, \ldots, r$, we see P_1 can use Eq. (3.2.14) to compute $f_j(x,y)$ over D. Hence, we have $\pi' \in \Pi_{prat}(\vec{f}; D)$. **Q.E.D.**

It should be noted that our proof can be easily extended to the case of computing more general continuously differentiable functions by using protocols whose messages are, for example, polynomial/rational functions.

3.2.4 Some New Lower Bounds

Throughout this subsection we assume that $f : D_f \mapsto C$ is a complex rational function, where $D_f \subset C^m \times C^n$ is the set of vectors (x,y) at which f is finite. In this context it is natural to consider "rational" protocols, in which the messages transmitted are rational functions of the input data (x,y).

We present two new methods for establishing lower bounds on the two–way communication complexity in this setting. The first method provides lower bounds on $C_{prat}(f; D)$ for any open subset $D \subset D_f$. The second method requires that $D = D_f$ but usually gives sharper lower bounds.

Computing the Product of Two Polynomials

Our first method (Theorems 3.2.7–3.2.8) exploits the fact that any rational protocol can be converted into a protocol in which the messages are rational functions of (x, y) and which uses at most twice as many messages.

Suppose that $f(x,y) = p(x,y)/q(x,y)$ where p and q are two relatively prime polynomials. In particular, $D_f = \{(x,y) \mid q(x,y) \neq 0\}$. Let $D \subset D_f$ be an open subset. Consider some rational protocol $\pi \in \Pi_{prat}(f; D)$ with r messages, where $r = C_{prat}(f; D)$ (cf. Subsection 1.1.1 of Chapter 1). Then, by Theorem 3.2.6, there exists a polynomial protocol $\pi' \in \Pi_{rpoly}(f; D)$ that uses $2r$ messages. Let $m_1, \ldots, m_{2r}$ be the message functions of the protocol π'. By the remark given at the end of Theorem 3.2.5, we see that

$$C_{prat}(f; D) \geq \frac{1}{2} C_{ppoly}(pg; D) = \frac{1}{2} C_{ppoly}(pg; C^{m+n})$$

and

$$C_{prat}(f; D) \geq \frac{1}{2} C_{ppoly}(qg; D) = \frac{1}{2} C_{ppoly}(qg; C^{m+n}),$$

where g is some nonzero polynomial. (The equalities in the above formulas follow from the fact that the domain of any purely polynomial protocols is the entire space C^{m+n}.) This shows that we can bound from below the communication complexity of f by bounding from below the communication complexity of pg or qg. The difficulty, however, is that the polynomial g is not known and we are forced to develop a bound which is valid for an arbitrary choice of g. Ideally, we would like to be able to say that if p has high communication complexity then the same is true for pg. Although this does not seem to be true in general, the following result makes a step in that direction.

Theorem 3.2.7 *Let $f, g \in C[x_1, \ldots, x_m, y_1, \ldots, y_n]$ be two nonzero polynomials which are relatively prime. Then,*

$$C_2(fg; C^{m+n}) \geq \max_{(x,y) \in V(f)} \operatorname{rank} H_{xy}(f)\Big|_{(x,y)} - 2,$$

where $V(f) = \{(x,y) \mid f(x,y) = 0\}$ is the zero set of the polynomial f.

Proof: By Theorem 3.2.1, we have

$$\begin{aligned} C_2(fg; C^{m+n}) &\geq \max_{(x,y)\in C^{m+n}} \operatorname{rank}\left(H_{xy}(fg)\right)|_{(x,y)} \\ &= \max_{(x,y)\in C^{m+n}} \operatorname{rank}\Big(f(x,y)H_{xy}(g)|_{(x,y)} + g(x,y)H_{xy}(f)|_{(x,y)} + \\ &\quad + (\nabla_x f(x,y))(\nabla_y g(x,y))^T + (\nabla_x g(x,y))(\nabla_y f(x,y))^T\Big) \\ &\geq \max_{(x,y)\in C^{n+m}} \operatorname{rank}\Big(f(x,y)H_{xy}(g)\Big|_{(x,y)} + g(x,y)H_{xy}(f)\Big|_{(x,y)}\Big) - 2 \\ &\geq \max_{(x,y)\in V(f)} \operatorname{rank}\Big(g(x,y)H_{xy}(f)\Big|_{(x,y)}\Big) - 2. \end{aligned}$$

Choose some $(x_0, y_0) \in V(f)$ such that

$$\operatorname{rank}\Big(H_{xy}(f)\Big|_{(x_0,y_0)}\Big) = \max_{(x,y)\in V(f)} \operatorname{rank} H_{xy}(f)\Big|_{(x,y)} = r.$$

Then, there exists a submatrix M of size $r \times r$ embedded in $H_{xy}(f)$, which is nonsingular at (x_0, y_0). We view this submatrix as a function of (x, y) and we consider its determinant $\det(M)$ which is a polynomial in (x, y). We have just shown that $V(f)$ is not contained in $V(\det(M))$. In other words, if we write f as a product of irreducible polynomials, then at least one of the irreducible factors of f, call it f_1, does not divide $\det(M)$. But since f and g are relatively prime, it follows that f_1 does not divide g either. We conclude that f_1 does not divide $g \cdot \det(M)$. We now claim that $V(f) \not\subset V(g \cdot \det(M))$. If the claim is not true, then $V(f) \subset V(g \cdot \det(M))$. Hilbert's Nullstellensatz applies and shows that $(g \cdot \det(M))^k = fh$ for some positive integer k and some polynomial h. By the unique factorization property, we see that the irreducible polynomial f_1 would have to be a factor of either g or $\det(M)$, which is a contradiction and establishes our claim.

Since $V(f) \not\subset V(g \cdot \det(M))$, there exists some $(x^*, y^*) \in V(f)$ such that $g(x^*, y^*)\det(M)|_{(x^*,y^*)} \neq 0$. Consequently,

$$\begin{aligned} \max_{(x,y)\in V(f)} \operatorname{rank}\left(g(x,y)H_{xy}(f)|_{(x,y)}\right) &\geq \operatorname{rank}\left(g(x^*,y^*)H_{xy}(f)|_{(x^*,y^*)}\right) \\ &= r \\ &= \max_{(x,y)\in V(f)} \operatorname{rank}\Big(H_{xy}(f)\Big|_{(x,y)}\Big). \end{aligned}$$

which completes the proof of the theorem. **Q.E.D.**

The above theorem states that if $\text{rank}H_{xy}(f)$ is large for some $(x,y) \in V(f))$ then fg also has large communication complexity for any polynomial g which is relatively prime to f. Unfortunately, Theorem 3.2.7 is not always sufficient for proving tight lower bounds for fg because there exist functions f for which $\text{rank}H_{xy}(f)$ is small for every $(x,y) \in V(f)$ even though $H_{xy}(f)$ has high rank when the restriction $(x,y) \in V(f)$ is removed. Below, we exhibit a polynomial f whose Hessian matrix vanishes identically on the set $V(f)$. In fact, it will be seen that f divides the polynomial $\det(H(f))$. Another important example will be seen in the next section.

Example[3]**:** Consider the polynomial

$$f(x_1,x_2,x_3,x_4) = x_4^2(x_3^2x_4^2 + (4x_2^3 - 6x_1x_2x_3)x_4 + 4x_1^3x_3 - 3x_1^2x_2^2)$$

By a simple calculation, we have

$$\begin{aligned}
\det\left(\frac{\partial^2 f}{\partial x_j \partial x_j}\right) &= 1440x_4^6(x_3^2x_4^2 - 6x_1x_2x_3x_4 + 4x_2^3x_4 + 4x_1^3x_3 - 3x_1^2x_2^2) \\
&\quad (x_3^2x_4^4 - 6x_1x_2x_3x_3^3 + 6x_2^3x_4^3 + 4x_1^3x_3^1x_4^2 - 9x_1^2x_2^2x_4^2 + 6x_1^4x_2x_4 - 2x_1^6) \\
&= 1440 \cdot f(x_1,x_2,x_3,x_4)x_4^4(x_3^2x_4^4 - 6x_1x_2x_3x_3^3 \\
&\quad +6x_2^3x_4^3 + 4x_1^3x_3^1x_4^2 - 9x_1^2x_2^2x_4^2 + 6x_1^4x_2x_4 - 2x_1^6).
\end{aligned}$$

Hence, the polynomial $\det(H(f))$ can be divided by f. Consequently, the Hessian matrix $H(f)$ cannot have full rank on the set $V(f)$. On the other hand, we have just seen that $\det(H(f))$ is a nonzero polynomial. Hence, $H(f)$ must be nonsingular at some point. This example illustrates that $\max_{C^n} \text{rank}H(f)$ is not always attained on the set $V(f)$.

The following is a result from algebraic geometry which gives a sufficient condition on f under which $H_{xy}(f)$ has high rank at some point belonging to $V(f)$.

Theorem 3.2.8 *Let $\hat{f}$ be a nonlinear homogeneous polynomial in n variables such that $\nabla\hat{f}(x) \neq 0$ for every $x \in V(\hat{f})$. Let the polynomial $f : C^{2n} \mapsto C$ be defined by $f(x,y) = \hat{f}(x+y)$. Then,*

$$\max_{(x,y)\in V(f)} \text{rank}\left(H_{xy}(f)\Big|_{(x,y)}\right) \geq n-1.$$

[3]This example was provided by Professor Steve Kleiman.

The proof of the above theorem can be found in [Z 83] and [K 79]. As an immediate consequence of above Theorems 3.2.7 and 3.2.8 we have the following:

Corollary 3.2.3 *Let f and $\hat{f}$ be as in Theorem 3.2.8 Then,*

$$C_2(f \cdot g; \mathcal{C}^{2n}) \geq n - 1$$

for any polynomial g which is relatively prime to f.

Unfortunately, the above corollary is not easy to apply, because the set $V(\hat{f})$ is usually hard to determine. Accordingly, the condition $\nabla \hat{f}(x) \neq 0$ on the set $V(\hat{f})$ cannot be easily tested. In fact, it seems a lot easier to just compute the rank of $H_{xy}(f)$ at a random point of $V(f)$ because $\max_{(x,y)\in V(f)} \text{rank} H_{xy}(f)|_{(x,y)}$ is attained at the majority of points on $V(f)$ (a Zariski open set of $V(f)$).

A Stronger Lower Bound

In the previous subsection, we have shown that lower bounds on the communication complexity of a rational function $f = p/q$ (p and q are relatively prime) can be obtained by developing lower bounds on the communication complexity of pg or qg, where g is an arbitrary nonzero polynomial. We now develop our second method for establishing lower bounds by exploiting the fact that if a protocol is to have domain the set D_f on which f is finite, then the polynomial g is not entirely arbitrary. We have shown earlier (Theorem 3.2.5) and the remark followed) that if f can be evaluated by a rational protocol with domain D_f, then there exists a polynomial g such that f is computed through the formula $f(x,y) = (pg)/(qg)$ and that $C_{prat}(f; D_f) \geq \frac{1}{2} C_{ppoly}(qg; D_f)$. The polynomial qg must certainly be nonzero at every point in the domain D_f of f because otherwise the expression $(pg)/(qg)$ will be meaningless for some $(x,y) \in D_f$. This additional constraint is used in an essential way in the following result.

Theorem 3.2.9 *Suppose that f is a rational function and that $f = p/q$, where $p, q \in \mathcal{C}[x_1, \ldots, x_m, y_1, \ldots, y_n]$ are relatively prime polynomials. If q is irreducible, then*

a)

$$C_{rat}(f; D_f) \geq \max_{(x,y)\in C^m \times C^n} \text{rank} H_{xy}(q)|_{(x,y)} - 1. \tag{3.2.15}$$

b)

$$C_{rat}(f; D_f) \geq \frac{1}{2} \max_{(x,y)\in C^m \times C^n} \text{rank} H_{xy}(p)|_{(x,y)} - \frac{3}{2}. \tag{3.2.16}$$

Proof: a) Consider a rational protocol for computing f on D_f that uses $r = C_{rat}(f; D_f)$ messages and let $m_1, \ldots, m_r : D_f \mapsto C$ be the corresponding message functions. We first consider the special case where each one of the message functions is a polynomial. Without loss of generality, we assume that the final evaluation of the function f is performed by processor P_1. By the definition of a rational protocol (cf. Chapter 1), there exists a rational function h such that $f(x,y) = h(x, m_1(x,y), \ldots, m_r(x,y))$ for all $(x,y) \in D_f$. Note that h can be expressed in the form

$$h(x, m_1(x,y), \ldots, m_r(x,y)) = \frac{h_1(x, m_1(x,y), \ldots, m_r(x,y))}{h_2(x, m_1(x,y), \ldots, m_r(x,y))},$$

where h_1 and h_2 are relatively prime polynomials. Let

$$h_2'(x,y) = h_2(x, m_1(x,y), \ldots, m_r(x,y)). \tag{3.2.17}$$

The functions $m_1, \ldots, m_r$ were originally defined on D_f. On the other hand, since they are polynomials they can be uniquely extended to polynomial functions on the entire of C^{m+n}. Furthermore, the representation (3.2.17) must be also valid over C^{m+n} and this implies that $C_{rpoly}(h_2'; C^{m+n}) \leq r$. We now notice that we must have $h_2'(x,y) \neq 0$ for all $(x,y) \in D_f$, because the function h must be defined for all $(x,y) \in D_f$. Equivalently, $V(h_2') \subset V(q)$, where q is the denominator polynomial of f, assumed irreducible. Hilbert's Nullstellensatz shows that $q^k = h_2' g$, for some polynomial g and some positive integer k. We factor the polynomial $h_2' g$ as a product of irreducible factors. Since q is irreducible, it follows that each one of these factors must be equal to q. We conclude that $h_2' = cq^K$ for some nonzero constant $c \in C$ and some positive integer K, and therefore $r \geq C_{rpoly}(h_2'; C^{m+n}) = C_{rpoly}(q^K; C^{m+n})$.

We now consider the case where there exists some i such that the message function m_i is not a polynomial and let us choose, in particular, the first such index i. Suppose, without loss of generality, that $i \in T_{1\to 2}$. We have $m_i(x,y) =$

$\hat{m}_i(x, m_1(x,y), \ldots, m_{i-1}(x))$ for some rational function $\hat{m}_i$ and each one of the functions $m_1, \ldots, m_{i-1}$ is a polynomial. We write $\hat{m}_i$ in the form $\hat{m}_i(x,y) = h_1(x,y)/h_2(x,y)$, where h_1 and h_2 are relatively prime polynomials. We now repeat the argument of the preceding paragraph. Since the domain of the protocol is all of D_f, it follows that $V(h_2) \subset V(q)$ and $h_2 = cq^K$ for some nonzero constant $c \in C$ and some positive integer K. Furthermore, it is clear that h_2 can be expressed as a polynomial function of $m_1(x,y), \ldots, m_{i-1}(x,y)$ which implies that $r > i-1 \geq C_{rpoly}(h_2'; C^{m+n}) = C_{rpoly}(q^K; C^{m+n})$.

To summarize, we have shown that in both cases there exists a positive integer K for which $C_{rat}(f; D_f) = r \geq C_{rpoly}(q^K; C^{m+n})$. It now remains to derive a lower bound on $C_{rpoly}(q^K; C^{m+n})$. To this effect, we apply Theorem 3.2.1. We have

$$\begin{aligned}
C_{rpoly}(q^K; C^{m+n}) &\geq C_2(q^K; C^{m+n}) \\
&\geq \max_{(x,y)\in C^{m+n}} \text{rank} H_{xy}(q^K)|_{(x,y)} \\
&= \max_{(x,y)\in C^{m+n}} \text{rank} \Big(Kq^{K-1}(x,y) H_{xy}(q)|_{(x,y)} + \\
&\quad + K(K-1) q^{K-2}(x,y) (\nabla_x q(x,y)) (\nabla_y q(x,y))^T \Big) \quad (3.2.18) \\
&\geq \max_{(x,y)\in C^{m+n}} \text{rank} \Big(Kq^{K-1}(x,y) H_{xy}(q)|_{(x,y)} \Big) - 1 \quad (3.2.19) \\
&\geq \max_{(x,y)\in C^{m+n}} \text{rank} H_{xy}(q)|_{(x,y)} - 1, \quad (3.2.20)
\end{aligned}$$

Here the equality (3.2.18) is a simple calculation and the next step (3.2.19) is due to the fact $(\nabla_x q)(\nabla_y q)^T$ has rank at most 1. The last step is obtained as follows. The set $\{(x,y) \mid q(x,y) \neq 0\}$ is a Zariski open set. Furthermore, the maximum rank of $H_{xy}(q)$ is attained at the set of points where the determinant of a suitable submatrix of $H_{xy}(q)$ does not vanish and is also a Zariski open set. Since every two nonempty Zariski open sets have nonempty intersection (Theorem 3.1.2), it suffices to consider a vector (x,y) in the intersection of these two sets.

b) Let (x,y) be an arbitrary vector of D_f. Note that

$$\begin{aligned}
H_{xy}\left(\frac{p}{q}\right) &= \frac{1}{q} H_{xy}(p) - \frac{p}{q^2} H_{xy}(q) - \frac{1}{q^2} (\nabla_x p)^T (\nabla_y q) \\
&\quad - \frac{1}{q^2} (\nabla_x q)^T (\nabla_y p) + \frac{2p}{q^3} (\nabla_x q)^T (\nabla_y q) .
\end{aligned}$$

By evaluating the rank of both sides at (x,y) and noticing that both $(\nabla_x p)^T (\nabla_y q)\,|_{(x,y)}$ and $(\nabla_x q)^T (\nabla_y q)\,|_{(x,y)}$ have rank at most 1, we see that

$$\mathrm{rank}H_{xy}(\frac{p}{q})|_{(x,y)} \geq \mathrm{rank}H_{xy}(p)|_{(x,y)} - \mathrm{rank}H_{xy}(q)|_{(x,y)} - 2.$$

Therefore,

$$\begin{aligned} C_{rat}(f;D_f) &\geq C_2(f;D_f) \\ &\geq \mathrm{rank}H_{xy}(\frac{p}{q})|_{(x,y)} \\ &\geq \mathrm{rank}H_{xy}(p)|_{(x,y)} - \mathrm{rank}H_{xy}(q)|_{(x,y)} - 2 \\ &\geq \mathrm{rank}H_{xy}(p)|_{(x,y)} - C_{rat}(f;D_f) - 3, \qquad \forall (x,y) \in D_f, \end{aligned}$$

where the last step follows from Eq. (3.2.15). After rearranging the above inequality, we see that

$$C_{rat}(f;D_f) \geq \frac{1}{2}\mathrm{rank}H_{xy}(p)|_{(x,y)} - \frac{3}{2}, \qquad \forall (x,y) \in D_f. \tag{3.2.21}$$

Since $\max_{(x,y)\in C^{m+n}} \mathrm{rank}H_{xy}(p)$ is attained at a Zariski open set and D_f is also a Zariski open set, by Theorem 3.1.2, there exists some vector $(x^*,y^*) \in D_f$ such that

$$\max_{(x,y)\in C^{m+n}} \mathrm{rank}H_{xy}(p) = \mathrm{rank}H_{xy}(p)|_{(x^*,y^*)}.$$

Now Eq. (3.2.16) becomes evident when one considers (3.2.21) at (x^*,y^*). **Q.E.D.**

3.2.5 An $\Omega(n^2)$ Lower Bound for Computing $[(x+y)^{-1}]_{11}$

Let x and y be $n \times n$ matrices. As an application of the results of Subsection 3.2.4, we consider the communication complexity of the function $f(x,y) = [(x+y)^{-1}]_{11}$ (the $(1,1)$th entry of $(x+y)^{-1}$) within the class of rational protocols. While Abelson's lower bound is only $\Omega(n)$, we derive a lower bound of n^2-1, which is almost equal to the obvious upper bound of n^2. In particular, this example will show that Abelson's bound can be far from tight.

We motivate our choice of the problem. The value of $[(x+y)^{-1}]_{11}$ can be thought of as the solution of the system of linear equations: $(x+y)u = b$, where

$b = (1,0,\ldots,0)$ and u is the unknown. Thus the problem under consideration captures the essential difficulties of a distributed solution of a system of the form $(x+y)u = b$, when x and y are possessed by different processors. Since the solution of linear systems of equations is the most basic problem in numerical computation, the problem we are studying is an interesting paradigm.

It is easy to see that n^2 messages would be needed if we had required that a particular processor, say P_1, should eventually evaluate all entries of the inverse matrix $(x+y)^{-1}$. (This is because P_1 could then invert $(x+y)^{-1}$ to obtain $x+y$ and use its knowledge of x to infer the value of y, and this is possible only if at least n^2 messages have been exchanged.) However, the fact that the evaluation of the whole inverse matrix $(x+y)^{-1}$ is hard does not imply that the computation of a particular entry is also difficult. In fact, we shall see that the derivation of tight bounds on the communication complexity of $[(x+y)^{-1}]_{11}$ is surprisingly hard. As a first indication, we show that Abelson's result (Theorem 3.2.1) gives only an $\Omega(n)$ lower bound.

Theorem 3.2.10 *Let $f(x,y) = [(x+y)^{-1}]_{11}$, where x, y are $n \times n$ matrices. Then*

$$\max_{(x,y)\in D_f} \text{rank} H_{xy}(f)|_{(x,y)} \leq 3n, \tag{3.2.22}$$

where $D_f = \{(x,y) \mid x+y$ is invertible or $\det(x+y) \neq 0\}$.

Proof: Let us fix a pair $p = (x_0, y_0) \in D_f$ of $n \times n$ matrices. We will show that the rank of $H_{xy}(f)|_p$ is at most $3n$. Let Δ_1, Δ_2 be two $n \times n$ perturbation matrices. We consider the Taylor series expansion of f at the point p:

$$\begin{aligned} f(x_0+\Delta_1, y_0+\Delta_2) &= \left[((x_0+y_0)-(\Delta_1+\Delta_2))^{-1}\right]_{11} \\ &= \left[(x_0+y_0)^{-1}\right]_{11} + \left[(x_0+y_0)^{-1}(\Delta_1+\Delta_2)(x_0+y_0)\right]_{11} \\ &\quad + \left[(x_0+y_0)^{-1}\left((\Delta_1+\Delta_2)(x_0+y_0)\right)^2\right]_{11} + \cdots \end{aligned}$$

Notice that the value of $H_{xy}(f)|_p$ is completely determined by the second order terms of this expansion. Thus, if we let

$$g(\Delta_1, \Delta_2) = [(x_0+y_0)^{-1}\left((\Delta_1+\Delta_2)(x_0+y_0)\right)^2]_{11},$$

then $H_{xy}(f)|_{(x_0,y_0)} = H_{\Delta_1\Delta_2}(g)|_{(0,0)}$. Therefore, we only need to show that $\text{rank}H_{\Delta_1\Delta_2}(g)|_{(0,0)} \leq 3n$. We will present a two–way polynomial protocol for computing g that only uses $3n$ messages.

Notice that as far as the computation of g is concerned, the matrices x_0, y_0 are constant and the matrices Δ_i $(i = 1, 2)$ are the inputs. Let $e = (1, 0, \ldots, 0)^T$. The protocol proceeds as follows.

1. Processor P_1 sends the vector $\Delta_1(x_0 + y_0)e$ to processor P_2 (n messages).
2. Processor P_2 computes $(\Delta_1+\Delta_2)(x_0+y_0)e$ and sends the following two vectors ($2n$ messages) to P_1:

$$(\Delta_1 + \Delta_2)(x_0 + y_0)e$$

and

$$\Delta_2(x_0 + y_0)(\Delta_1 + \Delta_2)(x_0 + y_0)e.$$

3. Once processor P_1 receives these messages, it can use its knowledge of Δ_1 to evaluate $((\Delta_1 + \Delta_2)(x_0 + y_0))^2 e$. It follows that

$$g(\Delta_1, \Delta_2) = [(x_0 + y_0)^{-1}((\Delta_1 + \Delta_2)(x_0 + y_0))^2]_{11}$$

can also be evaluated by P_1.

By Abelson's result (Theorem 3.2.1), we see that for any open set D containing $(0, 0)$, we have

$$\text{rank}H_{\Delta_1\Delta_2}(g)|_{(0,0)} \leq C_{rpoly}(g; D) \leq 3n,$$

which completes the proof. **Q.E.D.**

Let D_f be the set of all $(x, y) \in C^{n^2} \times C^{n^2}$ at which the rational function $f(x, y) = [(x+y)^{-1}]_{11}$ is well-defined. Clearly, D_f is the same as the set of all (x, y) such that $\det(x + y) \neq 0$. Our main result is the following.

Theorem 3.2.11

$$C_{rat}(f; D_f) \geq n^2 - 1. \tag{3.2.23}$$

The proof is based on two lemmas[4]:

Lemma 3.2.1 *The polynomial* $g(x,y) = \det(x+y)$ *is irreducible.*

Lemma 3.2.2 *Suppose that* $n > 1$ *and let* $g(x,y) = \det(x+y)$. *Then the rank of* $H_{xy}(g)$ *evaluated at* $(I,0)$ (I *is the identity matrix*) *is* n^2.

Once these two lemmas are proved, the desired result is obtained as follows. If $n = 1$, then Eq. (3.2.23) holds trivially. For $n > 1$, we have $f(x,y) = \det_{11}(x+y)/\det(x+y) = \det_{11}(x+y)/g(x,y)$, where $\det_{11}(x+y)$ is the cofactor of the (1,1)-th entry of $x+y$. It is seen that $g(x+y)$ does not divide $\det_{11}(x+y)$, because otherwise $[(x+y)^{-1}]_{11}$ would be a polynomial in the entries of x and y, which is easily shown not to be the case. Since g is irreducible (Lemma 3.2.1), we conclude that the polynomials $\det_{11}(x+y)$ and $g(x,y)$ are relatively prime. Then, Theorem 3.2.9 applies and shows that

$$C_{rat}(f; D_f) \geq \max_{(x,y)\in C^{2n^2}} H_{xy}(g)|_{(x,y)} - 1 \geq n^2 - 1,$$

where the last inequality has made use of Lemma 3.2.2. Thus, it only remains to prove the two lemmas.

Proof of Lemma 6.1: If $n = 1$, then $g(x,y) = x + y$, which is obviously an irreducible polynomial. For $n > 1$, we assume, in order to derive a contradiction, that $f(x,y) = A(x,y)B(x,y)$ where A, B are nonconstant polynomial functions of the entries of x, y. Let x_{ij} (respectively, y_{ij}) denote the (i,j)-th entry of x (respectively, y). Let us restrict x and y by letting $x_{1i} = -y_{1i} = 1, i = 2, \ldots, n$. With such a restriction, f, A, and B can be expressed as polynomials $\hat{f}$, $\hat{A}$, and $\hat{B}$, respectively, of the unrestricted variables. Note that

$$\hat{f}(x,y) = (x_{11} + y_{11})\det_{11}(x+y) = \hat{A}(x,y)\hat{B}(x,y).$$

By the unique factorization property of polynomials, we see that $(x_{11}+y_{11})$ must be a factor of either $\hat{A}(x,y)$ or $\hat{B}(x,y)$. Since $\det(x+y)$ is a linear function of $x_{11}+y_{11}$,

[4][VW 53] contains as an exercise the fact that $\det(x)$ is irreducible. It is easily seen that Lemma 3.2.1 implies this fact, but the converse is not true.

we conclude that x_{11}, y_{11} appear together in either $\hat{A}$ or $\hat{B}$, but not in both. It then follows that x_{11}, y_{11} appear together in either $A(x,y)$ or $B(x,y)$, but not in both. Repeating our argument for all (i,j) $(1 \leq i,j \leq n)$, we see that either x_{ij} and y_{ij} both appear only in $A(x,y)$ or they both appear only in $B(x,y)$. Therefore the set $\{(i,j),\ i,j = 1,2,\ldots,n\}$ can be partitioned into two subsets R_1, R_2 (with R_1 being nonempty) such that $A(x,y)$ depends only on the entries x_{ij}, y_{ij} with $(i,j) \in R_1$ and $B(x,y)$ depends only on the entries x_{ij}, y_{ij} with $(i,j) \in R_2$. Let us express each one of the polynomials A and B as a sum of products and then carry out the cross–multiplications to expand $A(x,y)B(x,y)$ as a sum of products. Since A and B depend on different entries, it is seen that this expansion leads to no cancellations. Hence, if (i,j) is in R_1 then (i,k) and (k,j), $k = 1,\ldots n$, also belong R_1, since otherwise there would be a term in the expansion of $A(x,y)B(x,y) = \det(x+y)$ with two entries from the same row or column. This implies that all of the entries must be in R_1, and R_2 is empty. Consequently, $B(x,y)$ is a constant polynomial, which contradicts our original assumption. **Q.E.D.**

Proof of Lemma 6.2: An easy calculation yields

$$\frac{\partial^2 g}{\partial x_{ij} \partial y_{lm}}\Big|_{(I,0)} = \begin{cases} 1 & \text{if } i = j,\ l = m \text{ and } i \neq l \\ -1 & \text{if } i = m,\ j = l \text{ and } i \neq l \\ 0 & \text{otherwise.} \end{cases}$$

Thus, if the rows and columns of $H_{xy}(g)|_{(I,0)}$ are suitably rearranged, the matrix $H_{xy}(g)|_{(I,0)}$ has the structure shown in Figure 3.1. It is not hard to see that this matrix is nonsingular and therefore has rank n^2.**Q.E.D.**

We would like to be able to strengthen Theorem 3.2.11 in a number of directions. First, Theorem 3.2.11 refers to the computation of $[(x+y)^{-1}]_{11}$, where x, y are complex matrices. This does not lead to a lower bound when we restrict x and y to be real, even though this is the case of main practical interest. A related deficiency is that the lower bound applies only to protocols whose domain is equal to all of D_f. It would be interesting to know whether the communication complexity of the problem can be reduced by an order of magnitude when we restrict to real matrices, or if we only consider the evaluation of f in an open set of real matrices. We conjecture that this is not the case, but we are not aware of any proof technique

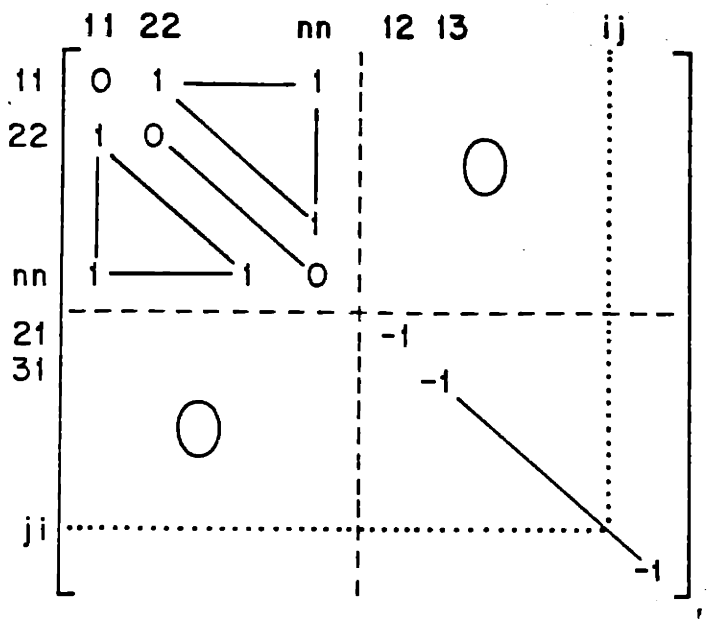

Figure 3.1: $H_{xy}(g)|_{(I,0)}$ after suitable rearrangement of the rows and columns.

that could lead to such a result.

One possible approach for proving a stronger lower bound is based on Theorem 3.2.7 of Section 5. This result shows that an $\Omega(n^2)$ lower bound will be established if we manage to find a pair (x,y) of matrices such that $g(x,y) = 0$ and $\text{rank}H_{xy}(g)|_{(x,y)} = \Omega(n^2)$, where $g(x,y) = \det(x+y)$. Unfortunately, the determinant function is particularly nasty in that respect, as is shown in the next theorem.

Theorem 3.2.12 *Let $g(x,y) = \det(x+y)$, where x and y are $n^2 \times n^2$ matrices. Then*

a) *There exists some nonzero constant c^* such that $\det(H_{xy}(g)) = c^* \cdot g^{n^2-2}$. In particular, the matrix $H_{xy}(g)$ has full rank at each point $(x,y) \in C^{n^2} \times C^{n^2}$ such that $g(x,y) \neq 0$.*

b) $\max_{(x,y)\in V(g)} \text{rank}H_{xy}(g) \leq 3n+3$.

Proof: a) Let $\bar{g}(z) = \det(z)$ and let $H(\bar{g})$ be its Hessian matrix. Then the matrix $H(\bar{g})$ evaluated at $z = x+y$ is the same as $H_{xy}(g)$. Hence, part a) will be evident

once we show that $H(\bar{g})$ has full rank at each point z such that z is invertible and that $\det(H(\bar{g})) = c^* \cdot \bar{g}^{n^2-2}$. By Lemma 3.2.2, the rank of $H(\bar{g})$ at the point $z = I$ is n^2. For other points, say $z = A$, where A is an invertible matrix, we are going to perform a nonsingular linear transformation of the variables of $\bar{g}$ so that A gets mapped to I. As a consequence, the rank of $H(\bar{g})$ at the point $z = A$ will be the same as the rank of $H(\bar{g})$ at $z = I$ which is equal to n^2. More specifically, we let the new variables be z' and therefore we have $z' = A^{-1}z$. We can rewrite it in a more standard form

$$(z'_{11}, \ldots, z'_{1n}, z'_{21}, \ldots, z'_{n1}, \ldots, z'_{nn})^T = B(z_{11}, \ldots, z_{1n}, z_{21}, \ldots, z_{n1}, \ldots, z_{nn})^T \quad (3.2.24)$$

where B is a matrix of size $n^2 \times n^2$ which, of course, is uniquely determined by A. Since one can also write z as a linear combination of z' in a unique way ($z = Az'$), it follows that B must be nonsingular.

Let us denote the elements of B by $(b_{ij,lm})$ and let also $g'(z') = \det(z')$. Noticing that $z'_{ij} = \sum_{l,m} b_{ij,lm} z_{lm}$ and that $\bar{g}(z) = \det(Az') = \det(A)\det(z')$, we see that

$$\frac{\partial \bar{g}}{\partial z_{ij}} = \det(A) \sum_{i',j'} b_{i'j',ij} \frac{\partial g'}{\partial z'_{i'j'}}$$

and

$$\frac{\partial^2 \bar{g}}{\partial z_{ij} \partial z_{lm}} = \det(A) \sum_{i',j',l',m'} b_{i'j',ij} b_{l'm',lm} \frac{\partial^2 g'}{\partial z_{i'j'} \partial z'_{l'm'}}, \qquad \forall 1 \le i,j,l,m \le n^2.$$

By writing the above relation in matrix form, we have

$$H(\bar{g}) = \det(A) B^T H(g') B. \quad (3.2.25)$$

Since B is nonsingular, it follows from Eq. (3.2.25) that $\text{rank}H(\bar{g})|_A = \text{rank}H(g')|_I = n^2$.

By taking the determinant of both sides of Eq.(3.2.25), we see that

$$\begin{aligned} \det H(\bar{g})|_{z=A} &= \det H(g')|_I [\det(A)]^{n^2} [\det(B)]^2 \\ &= c \cdot \bar{g}(A)^{n^2} [\det(B)]^2, \end{aligned} \quad (3.2.26)$$

where $c = \det H(g')|_I$ which is nonzero (Lemma 3.2.2).

We now show that $[\det(B)]^{-1}$ is a polynomial with variables being the entries of A. Let us denote the entries of A by a_{ij} $(i,j \leq n)$. Then, by comparing $z = Az'$ with $(z_{11},\ldots,z_{nn}) = B^{-1}(z'_{11},\ldots,z'_{nn})$ (see Eq.(3.2.24), we see that the entries of B^{-1} are linear functions of the a_{ij}'s. Hence, $[\det(B)]^{-1} = \det(B^{-1})$ remains a polynomial of a_{ij}'s. Since $\det H(\bar{g})|_{z=A}$ is a nonzero polynomial in a_{ij}'s, by Eq.(3.2.26), we see that the $\bar{g}(A)^{n^2}$ can be factored as a product of $[\det(B)]^{-1}$ times $\det H(\bar{g})|_{z=A}$. By checking the degrees of the corresponding polynomials and using the unique factorization property and the fact that $g(A)$ is irreducible (Lemma 3.2.1), we conclude that $[\det(B)]^{-1} = c'g(A)$ for some nonzero constant c'. Substituting this into Eq.(3.2.26), we obtain that $\det H(\bar{g})|_{z=A} = c^* g(A)^{n^2-2}$, where c^* is some new constant and $c^* \neq 0$. This completes the proof of part a).

b) We prove this indirectly. Let $h(x,y) = [(x+y)^{-1}]_{11}$. By Cramer's rule, we see that $h(x,y) = g_1(x,y)/g(x,y)$, where $g_1 = \det_{11}(x+y)$ which is the $(1,1)$–th cofactor of $\det(x+y)$. As shown in Lemma 3.2.22, the rank of $H_{xy}(h)$ is at most $3n$ at the points where $g(x,y) \neq 0$. By a straightforward calculation, we have

$$\begin{aligned} H_{xy}(h) &= \frac{1}{g}H_{xy}(g_1) - \frac{1}{g^2}(\nabla_x g_1)(\nabla_y g)^T - \frac{1}{g^2}(\nabla_x g)(\nabla_y g_1)^T \\ &\quad + \frac{2g_1}{g^3}(\nabla_x g)(\nabla_y g)^T - \frac{g_1}{g^2}H_{xy}(g), \end{aligned} \tag{3.2.27}$$

where $(\nabla_x g), (\nabla_y g)$ are $n^2 \times 1$ matrices formed by the partial derivatives of g with respect to variables x and y. The matrices $(\nabla_x g_1), (\nabla_y g_1)$ are defined similarly.

Note that both $(\nabla_x g_1)(\nabla_y g)^T$ and $(\nabla_x g)(\nabla_y g_1)^T$ are matrices of rank at most 1. Therefore, from Eq.(3.2.27), we see that if (x,y) is a point satisfying $g_1(x,y) = 0$, $g(x,y) \neq 0$, then there holds

$$\mathrm{rank}H_{xy}(h) \geq \mathrm{rank}H_{xy}(g_1) - 2.$$

Thus, we have

$$\mathrm{rank}H_{xy}(g_1) \leq \mathrm{rank}H_{xy}(h) + 2 \leq 3n + 2. \tag{3.2.28}$$

In other words, we have shown that the maximum rank (evaluated over the set of points (x,y) satisfying $g_1(x,y) = 0$ and $g(x,y) \neq 0$) of $H_{xy}(g_1)$ is at most $3n+2$. That is, on a Zariski open set of $V(g_1)$ (in the induced topology) the rank of $H_{xy}(g_1)$ is at most $3n+2$. However, by the fact that $\max_{V(g_1)} \mathrm{rank}H_{xy}(g_1)$ is attained also

on a Zariski open set of $V(g_1)$ and that any two Zariski open sets of $V(g_1)$ must overlap (Theorem 3.1.2), we conclude that $\max_{V(g_1)} \text{rank} H_{xy}(g_1)$ is at most $3n+2$. Here the use of Theorem 3.1.2 is justified since $V(g_1)$ is clearly an irreducible closed subset of C^n. It follows that $\max_{V(g)} \text{rank} H_{xy}(g) \leq 3n+3$. **Q.E.D.**

Finally, let us mention that an $\Omega(n^2)$ bound can also be obtained for the special case where x and y are restricted to be symmetric matrices. The proof is similar to the proof of Theorem 3.2.11.

3.3 Communication Complexity of the Integration Operator

In this section, we use Abelson's result to study the communication complexity of integration operator defined on the space $L^2[0,1]$ (the space of Lebesque measurable functions whose square is integrable over $[0,1]$). More precisely, we consider the operator I defined as:

$$I(f_1, f_2) = \int_0^1 f_1 f_2 dx, \qquad f_1, f_2 \in L^2[0,1]. \tag{3.3.1}$$

Let f_1, $f_2 : [0,1] \mapsto \Re$ be two functions in $L^2[0,1]$. Let there be two processors, say P_1 and P_2, who wish to compute the value $I(f_1, f_2)$. But processor P_i has access to the function f_i $(i = 1, 2)$ only. They exchange real–valued messages according to some protocol π, until one of the processors can evaluate $I(f_1, f_2)$. The notion of a communication protocol is defined in a manner similar to the one given in Subsection 1.1.1, except that the notion of differentiablity has to be adjusted since $D_x = L^2[0,1]$ has infinite dimension and the mappings m_i, $\hat{m}_i$, h_j (cf. Subsection 1.1.1) are real–valued functionals. There are several ways to define a derivative for a functional $f : L^2[0,1] \mapsto \Re$. In what follows, we will adopt the notion of Frechet derivative and say that a functional f is differentiable if and only if its Frechet derivative exists. The Frechet derivative is a natural generalization of the usual notion of derivative in a finite dimensional space and it is defined as follows:

Definition 3.3.1 *Let $f : M \mapsto \Re$ be a functional defined on some Hilbert space M. We say f is twice differentiable at $x \in M$, if there exist a continuous linear functional $L : M \mapsto \Re$ and a continuous bounded linear operator $A : M \mapsto M$ such that*

$$f(x+h) = f(x) + L(h) + (Ah, h) + o(||h||^2), \qquad \text{as } ||h|| \longrightarrow 0,$$

where $(\cdot, \cdot)$ denotes the inner product of M. L is called the Frechet derivative of f at x.

Having defined the notion of differentiability, we can now define the class of protocols for computing $I(f_1, f_2)$, in which the mappings m_i, $\hat{m}_i$ and h are continuously

twice differentiable functionals. We use the notation $\Pi_2(I; L^2[0,1] \times L^2[0,1])$ to denote such class of protocols. We are interested in knowing whether there exists a communication protocol $\pi \in \Pi_2(I; L^2[0,1] \times L^2[0,1])$ for computing $I(f_1, f_2)$ that uses only finite number of real–valued messages. In what follows, we show that the answer is negative.

By the basic L^2 space theory ([Ru 76]), there exists an orthonormal basis $\{g_i\}_{i=1}^{\infty}$ such that every function in $L^2[0,1]$ can be written as a weighted sum of the g_i's. The inner product of $L^2[0,1]$ is defined to be $I(f_1, f_2)$, for every f_1, $f_2 \in L^2[0,1]$.

For each positive integer n, let S_n be the linear subspace of $L^2[0,1]$ which is spanned by the vectors $\{g_1, \ldots, g_n\}$. Let f_1, f_2 be two arbitrary functions in S_n. Then, we can write

$$\begin{aligned} f_1 &= a_1 g_1 + a_2 g_2 \ldots + a_n g_n, \\ f_2 &= b_1 g_1 + b_2 g_2 \ldots + b_n g_n, \end{aligned}$$

where $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n$ are two arbitrary vectors in $\Re^n$. Since $\{g_i\}_{i=1}^{\infty}$ is an orthonormal basis, we see that

$$\begin{aligned} I(f_1, f_2) &= \int_0^1 (a_1 g_1 + \ldots + a_n g_n)(b_1 g_1 + \ldots + b_n g_n) dx \\ &= a_1 b_1 + a_2 b_2 + \ldots + a_n b_n. \end{aligned}$$

Now, we restrict the protocol π and the functional I to the subspace S_n and denote it by π_n, I_n. Since the subspace S_n is isomorphic to the Euclidean space $\Re^n$, we can view the functional I_n as a function $g_n : \Re^n \times \Re^n \mapsto \Re$ defined as

$$g_n(a_1, \ldots, a_n, b_1, \ldots, b_n) = \sum_{i=1}^{n} a_i b_i.$$

We consider the problem of computing g_n over $\Re^n \times \Re^n$ using the protocol π_n. By assumption, processor P_1 (respectively, P_2) knows the function f_1 (respectively, f_2), which implies that for the restricted problem processor P_1 knows the values of $a_1, \ldots, a_n$ (respectively, $b_1, \ldots, b_n$). Furthermore, using the Definition 3.3.1, we see that the twice differentiability of π quarantees that $\pi_n \in \Pi_2(g_n; \Re^n \times \Re^n)$. In light of Abelson's result (Theorem 3.2.1), $C_2(g_n; \Re^n \times \Re^n) = n$. Consequently, we have

$$\max_{f_1, f_2 \in S_n} r(I(f_1, f_2); \pi) = n,$$

where $r(I(f_1, f_2); \pi)$ denotes the number of messages used for computing $I(f_1, f_2)$ with protocol π. Since n and π are arbitrary, we have proved the following theorem:

Theorem 3.3.1

$$C_2(I; L^2[0,1] \times L^2[0,1]) = \infty,$$

where I is the integration operator defined by Eq. (3.3.1).

3.4 Computing Multiple Functions

Up to this point, we have only considered two–way protocols for computing a single function. In this section, we will look at more general protocols with which several functions $f_1, \ldots, f_s$ are to be computed. As we shall see later, when s is big and the functions $f_1, \ldots, f_s$ are simple (e.g., quadratic functions), the task of designing an efficient communication protocol can be converted to a combinatorial problem which can be solved using certain well–known tools from combinatorial optimization.

Let $f_j(x, y)$ $(j = 1, \ldots, s)$ be some functions defined on $\Re^{m+n}$, with $x \in \Re^m$ and $y \in \Re^n$. Suppose that two processors P_1 and P_2 are to compute these f_j's and processor P_1 (respectively, P_2) knows only the value of x (respectively, y) and the functional form of each f_j $(j = 1, \ldots, s)$. The two processors are to exchange messages, according to some protocol π, until each f_j can be computed either by processor P_1 or processor P_2 $(j = 1, \ldots, s)$. The notion of a two–way protocol for computing $f_1, \ldots, f_s$ is given in Subsection 1.1.1 of Chapter 1. Note that we do not require all of the f_j's be computed by the same processor and the computation is finished as soon as every f_j $(j = 1, \ldots, s)$ can be computed by either P_1 or P_2.

Suppose that π is a protocol for computing f_j's described by (see Subsection 1.1.1 of Chapter 1):

$$m_i(x, y) = \hat{m}_i\left(x, m_1(x, y), \ldots, m_{i-1}(x, y)\right), \quad \text{if } i \in T_{1\to 2}, \tag{3.4.1}$$

and

$$m_i(x, y) = \hat{m}_i\left(y, m_1(x, y), \ldots, m_{i-1}(x, y)\right), \quad \text{if } i \in T_{2\to 1}, \tag{3.4.2}$$

where the $\hat{m}_i$'s are some functions and $T_{1\to 2}$ (respectively, $T_{2\to 1}$) denotes the set of index i for which the ith message is sent by processor P_1 (respectively, by P_2). The functions $m_1, \ldots, m_r$ are called the message functions of π. Moreover, for each f_j, there exists an evaluation function h_j such that

$$f_j(x,y) = h_j(x, m_1(x,y), \ldots, m_r(x,y)), \tag{3.4.3}$$

if f_j is to be computed by processor P_1; if, on the other hand, f_j is to be computed by processor P_2, then

$$f_j(x,y) = h_j(y, m_1(x,y), \ldots, m_r(x,y)). \tag{3.4.4}$$

In the rest of this subsection, we will assume that each f_j $(j = 1, \ldots, s)$ is a polynomial and we consider the problem of computing the f_j's with a two–way purely polynomial protocol; equivalently, we assume that the functions m_i, $\hat{m}_i$, $i = 1, \ldots, r(\pi)$, and the functions h_j, $j = 1, \ldots, s$, are polynomials in x and y. Let $\Pi_{ppoly}(\vec{f})$ be the set of purely polynomial protocols and let $C_{ppoly}(\vec{f}; \Re^m \times \Re^n)$ denote the minimum number of messages needed for computing $f_1, \ldots, f_s$ with a protocol of $\Pi_{ppoly}(\vec{f})$. Furthermore, we let $\Pi^k_{ppoly} \subset \Pi_{ppoly}$ be the subclass of purely polynomial protocols for which the functions m_i, $\hat{m}_i$, h_j $(\forall i, j)$ are homogeneous polynomials of degree less than or equal to k. We let $C^k_{ppoly}(\vec{f}; \Re^m \times \Re^n)$ be the minimum number of messages needed for computing $f_1, \ldots, f_s$ with any protocol in Π^k_{ppoly}. Our next result says that the class of protocols Π^2_{ppoly} is almost as powerful as Π_{ppoly} for the problem of computing quadratic functions.

Theorem 3.4.1 *Let $f_i = x^T Q_i y + a_i^T x$, $i = 1, \ldots, s$, where each Q_i is an $m \times n$ constant matrix, $a_i \in \Re^m$, $b_i \in \Re^n$ are some constant vectors, and $x \in \Re^m$ and $y \in \Re^n$. Then*

$$C_{ppoly}(\vec{f}; \Re^m \times \Re^n) \leq C^2_{ppoly}(\vec{f}; \Re^m \times \Re^n) \leq 2C_{ppoly}(\vec{f}; \Re^m \times \Re^n).$$

Proof: The first inequality is obvious since $\Pi^2_{ppoly} \subset \Pi_{ppoly}$. To prove the second inequality, we will, by doubling the number of messages, transform any protocol $\pi \in \Pi_{ppoly}$ for computing $f_1, \ldots, f_s$ into a protocol $\pi' \in \Pi^2_{ppoly}$ which also computes the f_j's. In particular, let the message functions of π be $m_1, \ldots, m_r$. In what

follows, for any polynomial f, we will use $f^{(1)}$ and $f^{(2)}$ to denote its linear terms and quadratic terms.

If $i \in T_{1\to 2}$, then using Eq. (3.4.1) and the fact that $\hat{m}_i$, m_i are polynomials, we see that $m_i^{(1)}(x,y)$ is a linear combination of $m_1^{(1)}(x,y),\ldots,m_{i-1}^{(1)}(x,y)$ and x. In other words, we have

$$m_i^{(1)}(x,y) = c_{i1}m_1^{(1)}(x,y) + \cdots + c_{i,i-1}m_{i-1}^{(1)}(x,y) + c_i^T x, \tag{3.4.5}$$

where the vector c_i and the constants c_{ik} can be determined uniquely by the polynomial $\hat{m}_i$. Furthermore, by the fact that higher (i.e., more than 3rd) order terms of $m_k(x,y)$ $(k<i)$ do not affect $m_i^{(2)}(x,y)$, we then have

$$m_i^{(2)}(x,y) = \bar{m}_i\left(x, m_1^{(1)}(x,y), m_1^{(2)}(x,y),\ldots,m_{i-1}^{(1)}(x,y), m_{i-1}^{(2)}(x,y)\right), \tag{3.4.6}$$

where $\bar{m}_i$ is some polynomial whose degree is no more than 2. The case where $i \in T_{2\to 1}$ can be dealt similarly.

We now define the protocol π' as follows. π' has a total of $2r$ messages $m'_1,\ldots,m'_{2r}$ which are given by

$$m'_{2i-1} = m_i^{(1)}(x,y), \qquad m'_{2i} = m_i^{(2)}(x,y), \qquad i=1,\ldots,r.$$

If $i \in T_{1\to 2}$, then in light of Eqs. (3.4.5) and (3.4.6), we see that

$$m'_{2i-1}(x,y) = c_{i1}m'_1(x,y) + \cdots + c_{i,i-1}m'_{2i-2}(x,y) + c_i^T x,$$

and

$$m'_{2i}(x,y) = \bar{m}_i\left(x, m'_1(x,y),\ldots,m'_{2i-2}(x,y)\right),$$

which implies that $m'_{2i-1}(x,y)$ and $m'_{2i}(x,y)$ are functions of previous messages and x when $i \in T_{1\to 2}$. Similarly, we can show that the same holds when $i \in T_{2\to 1}$. Thus, π' is indeed a legitimate protocol.

It remains to show that π' is adequate for computing the f_i's. To see this, we consider Eq. (3.4.3). Since each f_j is a quadratic function and h_j is a polynomial, we see that $f_j(x,y)$ only depends on $m_i^{(1)}(x,y), m_i^{(2)}(x,y)$ $(i=1,\ldots,r)$ and x (assuming that f_j is computed by P_1). This means that there exists a polynomial $\bar{h}_j$ of degree no more than 2 such that

$$f_j(x,y) = \bar{h}_j\left(x, m'_1(x,y),\ldots,m'_{2r}(x,y)\right),$$

which proves that π' is indeed adequate for computing f_j's. Since for every i and j the polynomials $m_i', \bar{m}_i, \bar{h}_j$ have degree no more than 2, it follows that $\pi' \in \Pi^2_{ppoly}$. **Q.E.D.**

Theorem 3.4.1 says basically that we can restrict ourselves to the protocols of Π^2_{ppoly} without increasing too much the communication complexity. We can easily generalize Theorem 3.4.1 to the case of computing multiple polynomials with degrees less than or equal to k, where k is some positive integer, and show that

$$C_{ppoly}(\vec{f}; \Re^m \times \Re^n) \leq C^k_{ppoly}(\vec{f}; \Re^m \times \Re^n) \leq kC_{ppoly}(\vec{f}; \Re^m \times \Re^n).$$

We will now study in some more detail the quantity $C^2_{ppoly}(\vec{f}; \Re^m \times \Re^n)$ for the simple case where each Q_i is of rank 1.

Suppose that $f = x^T Q y$ is a quadratic function. Let $\pi \in \Pi^2_{ppoly}$ be a protocol for computing f, described by Eqs. (3.4.1)–(3.4.4). Without loss of generality, let us assume that the linear messages are functions of x or y alone. In fact, if $m_i(x, y) = c_i^T x + d_i^T y$ and $i \in T_{1\to 2}$ is a linear message of π, then we can replace m_i by the message $m_i' = c_i^T x$. This is because that upon receiving m_i' processor P_2 can compute $m_i = m_i' + d_i^T y$ and continue with protocol π. We can repeat this process until all the linear messages of π become linear functions of x or linear functions of y, while at the same time keep the number of messages unchanged. Thus, we can assume that the linear messages of π are $c_1^T x, \ldots, c_l^T x$ and $d_1^T y, \ldots, d_k^T y$. Let $f = x^T Q y$ be a quadratic function and $\text{rank}(Q) = 1$. By Eq. (3.2.3), we see that

$$f(x, y) = x^T U V^T y, \tag{3.4.7}$$

where $U \in \Re^m$ and $V \in \Re^n$.

Lemma 3.4.1 *Let f be given by Eq. (3.4.7). Suppose that π is a protocol for computing f whose linear messages are $c_1^T x, \ldots c_l^T x$ and $d_1^T y, \ldots, d_k^T y$. Then, either $U \in \text{span}\{c_1, \ldots, c_l\}$ or $V \in \text{span}\{d_1, \ldots, d_k\}$.*

Proof: Since all the messages of π are homogeneous polynomials of degree bounded by 2, we see that every quadratic function computable with linear messages $c_1^T x, \ldots, c_l^T x$ and $d_1^T y, \ldots, d_k^T y$ must be of the form:

$$c_1^T x g_1^T y + \cdots + g_l^T x c_l^T y + h_1^T x d_1^T y + \cdots + h_k^T x d_k^T y,$$

where $g_1, \ldots, g_l \in \Re^m$ and $h_1, \ldots, h_k \in \Re^n$. Hence, we can write

$$x^T U V^T y = c_1^T x g_1^T y + \cdots + c_l^T x g_l^T y + h_1^T x d_1^T y + \cdots + h_k^T x d_k^T y, \tag{3.4.8}$$

for some vectors $g_1, \ldots, g_l$ and $h_1, \ldots, h_k$. If both $U_i \notin \text{span}\{c_1, \ldots, c_l\}$ and $V_i \notin \text{span}\{d_1, \ldots, d_k\}$, then one can find some x and y such that $c_i^T x = 0$, $d_j^T y = 0$ for $i = 1, \ldots, l$, $j = 1, \ldots, k$ and $x^T U \neq 0$, $V^T y \neq 0$. Plugging this choice of x and y into Eq. (3.4.8), we see that the left hand side is nonzero while the right hand side is zero. Contradiction! **Q.E.D.**

At this point, we need to make a definition.

Definition 3.4.1 *Let (U_i, V_i) be pairs of vectors such that $U_i \in \Re^m$ and $V_i \in \Re^n$, $i = 1, \ldots, s$. Let $\mathcal{A}$ be a linear subspace of $\Re^m$ and $\mathcal{B}$ be a linear subspace of $\Re^n$. We say that $\mathcal{A}$ and $\mathcal{B}$ cover $\{(U_1, V_1), \ldots, (U_s, V_s)\}$ if for each i $(1 \leq i \leq s)$, either $U_i \in \mathcal{A}$ or $V_i \in \mathcal{B}$.*

Let $f_i(x, y) = x^T Q_i y$ $(i = 1, \ldots, s)$ be some quadratic functions and $\text{rank}(Q_i) = 1$. We can write f_i as $x^T U_i V_i^T y$ for some vectors U_i and V_i. As a corollary of Lemma 3.4.1, we have the following theorem.

Theorem 3.4.2 *Let $\mathcal{A}$ and $\mathcal{B}$ be some linear subspaces of $\Re^m$ and $\Re^n$ respectively such that $(\mathcal{A}, \mathcal{B})$ cover $\{(U_1, V_1), \ldots, (U_s, V_s)\}$. Then*

$$C^2_{ppoly}(\vec{f}; \Re^m \times \Re^n) \leq \dim\mathcal{A} + \dim\mathcal{B}, \tag{3.4.9}$$

and the equality can be achieved by some choice of $\mathcal{A}$ and $\mathcal{B}$.

Proof: Let $\mathcal{A}$ and $\mathcal{B}$ be some subspaces that cover $\{(U_1, V_1), \ldots, (U_s, V_s)\}$. Assume that $\{c_1, \ldots, c_l\}$ is a basis of $\mathcal{A}$ and $\{d_1, \ldots, d_k\}$ is a basis of $\mathcal{B}$. We define a protocol $\pi \in \Pi^2_{ppoly}$ as follows: $m_i(x, y) = c_i^T x$, for $i = 1, \ldots, l$, and $m_{i+l}(x, y) = d_i^T y$ for $i = 1, \ldots, k$. For each i, if $U_i \in \mathcal{A}$, then $U_i^T x$ can be computed from messages $m_1, \ldots, m_l$ and therefore f_i can be computed by P_2. Similarly, if $V_i \in \mathcal{B}$, then f_i can be computed by P_1, using messages $m_{l+1}, \ldots, m_{l+k}$. Since, for each i, (U_i, V_i) is covered by $(\mathcal{A}, \mathcal{B})$, it follows that f_i can be computed with message functions $m_1, \ldots, m_{l+k}$, which proves Eq. (3.4.9).

It remains to show that the equality of Eq. (3.4.9) can be attained by some linear subspaces $\mathcal{A}$ and $\mathcal{B}$. Let $\pi \in \Pi^2_{ppoly}$ be a protocol for computing the f_i's such that $r(\pi) = C^2_{ppoly}(\vec{f}; \Re^m \times \Re^n)$. As noted before, we can, without loss of generality, assume that the linear messages of π are of the form $c_1^T x, \ldots, c_l^T x$ and $d_1^T y, \ldots, d_k^T y$. Let $\mathcal{A}$ (respectively, $\mathcal{B}$) stand for the linear subspace of $\Re^m$ (respectively, $\Re^n$) spanned by the vectors $\{c_1, \ldots, c_l\}$ (respectively, $\{d_1, \ldots, d_k\}$). Since $f_i = x^T U_i V_i^T y$ is computed by π, by Lemma 3.4.1, we see that either $U_i \in \mathcal{A}$ or $V_i \in \mathcal{B}$. Hence $(\mathcal{A}, \mathcal{B})$ covers $\{(U_1, V_1), \ldots, (U_s, V_s)\}$. On the other hand, $r \geq l + k \geq \dim\mathcal{A} + \dim\mathcal{B}$. By Eq. (3.4.9), we see that $r = \dim\mathcal{A} + \dim\mathcal{B}$. **Q.E.D.**

Remark: We have actually shown that

$$C^1_{ppoly}(\vec{f}; \Re^m \times \Re^n) = C^2_{ppoly}(\vec{f}; \Re^m \times \Re^n) = \min\{\dim \mathcal{A} + \dim \mathcal{B}\},$$

where the minimum is taken over all linear subspaces $\mathcal{A}$ and $\mathcal{B}$ such that $(\mathcal{A}, \mathcal{B})$ covers $\{(U_1, V_1), \ldots, (U_s, V_s)\}$.

Theorem 3.4.2 provides a simple algebraic characterization of $C^2_{ppoly}(\vec{f}; \Re^m \times \Re^n)$, for the case when $f_i = x^T Q_i y$ and $\mathrm{rank}(Q_i) = 1$, $i = 1, \ldots, s$. Naturally, we ask the question of how to find a pair of linear subspaces $\mathcal{A}$ and $\mathcal{B}$ such that $(\mathcal{A}, \mathcal{B})$ covers $\{(U_1, V_1), \ldots, (U_s, V_s)\}$ and the value of $\dim \mathcal{A} + \dim \mathcal{B}$ is minimum. One can, in principle, try to evaluate $\dim\mathcal{A}(I) + \dim\mathcal{B}(I^c)$, for all linear subspaces of the form $\mathcal{A} = \mathrm{span}\{U_i; i \in I\}$ and $\mathcal{B} = \mathrm{span}\{V_i; i \in I^c\}$, where I is some subset of $\{1, \ldots, s\}$ and I^c denotes the complement of I. By Theorem 3.4.2, we see that $C^2_{ppoly}(\vec{f}; \Re^m \times \Re^n) = \min_I\{\dim\mathcal{A}(I) + \dim\mathcal{B}(I^c)\}$. However, when s is big, this process can take a long time since the total number of different I's is exponential in s. What we need is a polynomial time algorithm that can compute $C^2_{ppoly}(\vec{f}; \Re^m \times \Re^n)$. However, in general, the problem is unfortunately quite complex and we do not yet have a solution for it. Nonetheless, in a special case, a polynomial time algorithm does indeed exist.

Theorem 3.4.3 *Let $f_i = x^T U_i V_i^T y$. Suppose that $U_i \in \{c_1, \ldots, c_l\}$ and $V_i \in \{d_1, \ldots, d_k\}$, $i = 1, \ldots, s$, where $c_1, \ldots c_l$ and $d_1, \ldots, d_k$ are two sets of linearly independent vectors in $\Re^m$ and $\Re^n$ respectively. Then there exists a polynomial time algorithm for finding an optimal protocol in Π^2_{ppoly} that computes the f_i's.*

Proof: We create a bipartite graph $G = (N_1 \bigcup N_2, E)$. N_1 has l nodes $u_1, \ldots, u_l$, where each u_i corresponds to the vector c_i. Similarly, there are m nodes $v_1, \ldots, v_k$ in N_2 and the node v_j corresponds to the vector d_j. In addition, G has s edges defined as follows: $(u_i, v_j) \in E$ if and only if $U_t = c_i$ and $V_t = d_j$ for some t $(1 \leq t \leq s)$. We claim that the problem of finding a pair of linear subspaces $(\mathcal{A}, \mathcal{B})$ for covering the (U_i, V_i)'s such that $\dim \mathcal{A} + \dim \mathcal{B}$ is minimum can be reduced to the problem of finding a minimum node cover of G. Given any subset $I \subset \{1, \ldots, s\}$, consider the two subspaces $\mathcal{A}(I)$ and $\mathcal{B}(I^c)$. We let $N_1(\mathcal{A}(I)) = \{u_i |\ c_i \in \mathcal{A}(I)\}$ and $N_2(\mathcal{B}(I^c)) = \{v_j |\ d_j \in \mathcal{B}(I^c)\}$. By our construction, we see that $N_1(\mathcal{A}(I))$ and $N_2(\mathcal{B}(I^c))$ cover the edges of G if and only if $(\mathcal{A}(I), \mathcal{B}(I^c))$ covers $\{(U_1, V_1), \ldots, (U_s, V_s)\}$. Furthermore, since the vectors $c_1, \ldots, c_l$ (respectively, $d_1, \ldots, d_k$) are linearly independent, it follows that the cardinality of $N_1(\mathcal{A}(I))$ (respectively, $N_2(\mathcal{B}(I^c))$) is equal to $\dim \mathcal{A}(I)$ (respectively, $\dim \mathcal{B}(I^c)$). Conversely, suppose that $M_1 \subset N_1$ and $M_2 \subset N_2$ is an arbitrary node cover of G. Let

$$I = \{i | U_i = c_k \text{ for some } t \text{ s.t. } u_t \in M_1,\ 1 \leq i \leq s\}.$$

Then, it is easy to see that $(\mathcal{A}(I), \mathcal{B}(I^c))$ forms a cover for (U_i, V_i)'s. Moreover, $\dim \mathcal{A}(I) = |M_1|$ and $\dim \mathcal{B}(I^c) = |M_2|$. Therefore, we have just shown that the problem of finding the subspaces $\mathcal{A}$ and $\mathcal{B}$ such that $(\mathcal{A}, \mathcal{B})$ covers the (U_i, V_i)'s and $\dim \mathcal{A} + \dim \mathcal{B}$ is minimum is equivalent the problem of finding a minimum node cover for the graph G. Consequently, by Theorem 3.4.2, we see that $C^2_{ppoly}(\vec{f}; \Re^m \times \Re^n)$ is equal to the minimum cardinality node cover of G. It is well known (see [PS 82, pages 221–234]) that the minimum node cover problem for a bipartite graph is solvable in polynomial time by using a bipartite matching algorithm. **Q.E.D.**

3.5 A New Lower Bound Approach

In this section, we develop a new lower bound approach that is different from that of Abelson's. Our result is fairly intuitive, but surprisingly hard to prove. As opposed to Abelons's bound which uses only the second order derivatives of f (the function to be computed), our result has to do with the first order derivatives of f. We also point out that in certain situations our result can be far superior to that of Abelson's. The organization of this section is as follows. In Subsection 3.5.1, we establish some basic analytic properties of optimal protocols. In Subsection 3.5.2, we prove a general lower bound for the problem of computing an arbitrary continuously differentiable function f with protocols in which the message functions are continuously differentiable and the final evaluation function is continuous. To illustrate the power of our bound, in Subsection 3.5.3 we consider the case where f is given as a root z of the polynomial $\sum_{i=0}^{n-1}(x_i + y_i)z^i$ and obtain a lower bound of n, in contrast to the $\Omega(1)$ lower bound obtained by applying Abelson's result. In Subsection 3.5.4, we point out a situation where Abelson's bound gives rise to an optimal bound, whereas our result can only provide a lower bound of $\Omega(1)$. Thus, Abelson's result and our result are in general incomparable.

3.5.1 Analytical Properties of Optimal Protocols

Let $f(x, y)$ be some continuously differentiable function defined on $D_x \times D_y$, where D_x and D_y are open subsets of $\Re^m \times \Re^n$ respectively. We use the notation $\nabla_x f(x, y)$ and $\nabla_y f(x, y)$ to denote the m–dimensional (respectively, n–dimensional) vector whose components are the partial derivatives of f with respect to the components of x (respectively, y). For any open subsets $\overline{D}_x \subset D_x$ and $\overline{D}_y \subset D_y$, we let $\Pi_0(f; \overline{D}_x \times \overline{D}_y)$ be the class of two–way communication protocols for computing f on $\overline{D}_x \times \overline{D}_y$ whose message functions are continuously differentiable and the final evaluation function is continuous. We let

$$r = \min_{\overline{D}_x, \overline{D}_y} C_1(f; \overline{D}_x \times \overline{D}_y), \tag{3.5.1}$$

where the minimum is taken over all nonempty open subsets $\overline{D}_x, \overline{D}_y$ of D_x, D_y, respectively. In order to simplify notation, we will assume that the minimum is

attained with $\overline{D}_x = D_x$, $\overline{D}_y = D_y$. Thus, $r = C_1(f; D_x \times D_y)$. Let us consider a protocol that uses exactly r messages, described by (cf. Chapter 1)

$$m_i(x,y) = \hat{m}_i(x, m_1(x,y), \ldots, m_{i-1}(x,y)), \qquad \forall (x,y) \in D_x \times D_y, \text{ if } i \in T_{1\to 2}, \tag{3.5.2}$$

and

$$m_i(x,y) = \hat{m}_i(y, m_1(x,y), \ldots, m_{i-1}(x,y)), \qquad \forall (x,y) \in D_x \times D_y, \text{ if } i \in T_{2\to 1}, \tag{3.5.3}$$

where each m_i and $\hat{m}_i$ is a continuously differentiable function. Without loss of generality, we assume that the final evaluation of f is performed by processor P_1. Thus, there exists some continuous function h such that

$$f(x,y) = h(x, m_1(x,y), \ldots, m_r(x,y)), \quad \forall (x,y) \in D_x \times D_y. \tag{3.5.4}$$

We introduce some notations. Let $u = (x,y)$ and let $D = D_x \times D_y$. Let also $m(u) = (m_1(u), \ldots, m_r(u))$ and let $\nabla m(u)$ be the $(m+n) \times r$ matrix whose i-th column is the gradient vector $\nabla m_i(u)$, $i = 1, \ldots, r$. Define

$$k = \max_{u \in D} \operatorname{rank} [\nabla m(u)]. \tag{3.5.5}$$

Theorem 3.5.1 *For any optimal protocol, we have $k = r$.*

Proof: We show this by contradiction. Suppose that $r > k$. Consider the continuously differentiable mapping $m : D \mapsto \Re^r$, where $D = D_x \times D_y$ is an open set and $m(u) = (m_1(u), \ldots, m_r(u))$. We claim that $\nabla m_1(x,y)$ is not identically zero on the set D. Indeed, if this was the case, then $m_1(x,y)$ would be equal to a constant on the set D, and the first message in the protocol would be redundant. Thus, there would exist a protocol that uses $r-1$ messages, contradicting the definition of r. We can therefore apply Theorem A.1.1 in the appendix (with the correspondence $m \leftrightarrow F$, $D \leftrightarrow Q$, $r \leftrightarrow s$) to conclude that there exists some positive integer i and some continuously differentiable function g such that

$$m_{i+1}(u) = g\left(m_1(u), \ldots, m_i(u)\right), \qquad \forall u \in \overline{D}, \tag{3.5.6}$$

where $\overline{D}$ is some nonempty open subset of D. By taking a subset of $\overline{D}$ if necessary, we can assume that $\overline{D}$ is of the product form $\overline{D}_x \times \overline{D}_y$, where $\overline{D}_x$ and $\overline{D}_y$ are some open subsets of D_x and D_y respectively. Then, Eq. (3.5.6) would imply that the $(i+1)$-st message $m_{i+1}(x,y)$ is redundant for computing f over $\overline{D}_x \times \overline{D}_y$, which contradicts the definition of r (cf. Eq. (3.5.1)). **Q.E.D.**

Loosely speaking, Theorem 3.5.1 tells us that each message in an optimal protocol has to contain some "new information" and therefore the corresponding gradient vectors have to be linearly independent. Before we go on to the next theorem, we introduce some more notations. Let $\overline{D}_x \subset D_x$, $\overline{D}_y \subset D_y$ be nonempty open sets such that $\nabla m(u)$ has full rank for every $u \in \overline{D}_x \times \overline{D}_y$. (Such sets can be taken nonempty due to Theorem 3.5.1, and open due to the continuity of $\nabla m(u)$.) We use $\overline{D}$ as a short notation for $\overline{D}_x \times \overline{D}_y$. Furthermore, for any vector $c = (c_1, \ldots, c_r) \in \Re^r$, we let $c^i = (c_1, c_2, \ldots, c_i)$. Let also r_1 (respectively, r_2) be the number of messages sent by processor P_1 (respectively, P_2). In addition, we use the notation $[\nabla_x m_i(x,y); i \in T_{1\to 2}]$ to denote the $m \times r_1$ matrix whose column vectors are $\nabla_x m_i(x,y) = \left(\frac{\partial m_i}{\partial x_1}(x,y), \ldots, \frac{\partial m_i}{\partial x_m}(x,y)\right)$, $i \in T_{1\to 2}$. The $n \times r_2$ matrix $[\nabla_y m_i(x,y); i \in T_{2\to 1}]$ is defined similarly. As a refinement of Theorem 3.5.1, we have the following:

Theorem 3.5.2 *For any optimal protocol and $(x,y) \in \overline{D}$, there holds*

$$\text{rank}[\nabla_x \hat{m}_i(x, c^{i-1}); i \in T_{1\to 2}] = r_1,$$

and

$$\text{rank}[\nabla_y \hat{m}_i(y, c^{i-1}); i \in T_{2\to 1}] = r_2,$$

where $c = m(x,y)$.

Proof: By Theorem 3.5.1, we see that the matrix $\nabla m(x,y)$ has full rank (and its rank is equal to r) over the set $\overline{D}$. Notice that by possibly reindexing the columns of the matrix $\nabla m(x,y)$ we can write $\nabla m(x,y)$ in the form

$$\nabla m(x,y) = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

where $A_{11} = [\nabla_x m_i(x,y); i \in T_{1\to 2}]$ and $A_{22} = [\nabla_y m_i(x,y); i \in T_{2\to 1}]$. From Eqs. (3.5.2)–(3.5.3), it is easily seen that for each $i \in T_{2\to 1}$, there exists a continuously differentiable function M_i such that

$$m_i(x,y) = M_i\left(y, \{m_l(x,y) : l < i,\ l \in T_{1\to 2}\}\right), \qquad i \in T_{2\to 1}. \tag{3.5.7}$$

(In other words, a message sent by processor P_2 can be expressed as a function of y and the messages already received.) By differentiating Eq. (3.5.7), we obtain

$$\nabla_x m_i(x,y) = \sum_{l \in T_{1\to 2}} d_l(x,y) \nabla_x m_l(x,y), \qquad i \in T_{2\to 1} \tag{3.5.8}$$

where each $d_l(x,y)$ is a suitable scalar. Thus,

$$\nabla_x m_i(x,y) \in \text{span}\ \{\nabla_x m_l(x,y); i \in T_{2\to 1}\}, \qquad \forall (x,y) \in \overline{D},\ \forall l \in T_{1\to 2}.$$

This means that the columns of A_{12} belong to the span of the columns of A_{11} and therefore

$$\text{rank}\begin{bmatrix} A_{11} & A_{12} \end{bmatrix} = \text{rank}\,(A_{11}) \le r_1.$$

Similarly, one can show that

$$\text{rank}\begin{bmatrix} A_{21} & A_{22} \end{bmatrix} = \text{rank}\,(A_{22}) \le r_2.$$

On the other hand,

$$\begin{aligned} r &= r_1 + r_2 \\ &\ge \text{rank}\,(A_{11}) + \text{rank}\,(A_{22}) \\ &= \text{rank}\begin{bmatrix} A_{11} & A_{12} \end{bmatrix} + \text{rank}\begin{bmatrix} A_{21} & A_{22} \end{bmatrix} \\ &\ge \text{rank}\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \\ &= \text{rank}[\nabla m(x,y)] \\ &= r, \qquad \forall (x,y) \in \overline{D}. \end{aligned}$$

This implies that

$$\text{rank}(A_{11}) = \text{rank}[\nabla_x m_i(x,y); i \in T_{1\to 2}] = r_1$$

and

$$\text{rank}(A_{22}) = \text{rank}[\nabla_y m_i(x,y); i \in T_{2\to 1}] = r_2.$$

To show that $\text{rank}[\nabla_x \hat{m}_i(x, c^{i-1}); i \in T_{1\to 2}] = r_1$, we differentiate Eq. (3.5.2) to obtain

$$\nabla_x m_i(x,y) = \nabla_x \hat{m}_i(x, c^{i-1}) + \sum_{l=1}^{i-1} \frac{\partial \hat{m}_i}{\partial m_l}(x, c^{i-1}) \nabla_x m_l(x,y), \qquad \text{if } i \in T_{1\to 2}, \quad (3.5.9)$$

where $c = m(x,y)$ and $(x,y) \in \overline{D}$. Using Eq. (3.5.8), we see that $\sum_{l=1}^{i-1} \frac{\partial \hat{m}_i}{\partial m_l}(x, c^{i-1}) \nabla_x m_l(x,y)$ can be written as a linear combination of the vectors $\{\nabla_x m_l(x,y);\ l < i-1,\ l \in T_{1\to 2}\}$. Therefore, Eq. (3.5.9) shows that

$$[\nabla_x \hat{m}_i(x, c^{i-1}); i \in T_{1\to 2}] = [\nabla_x m_i(x,y); i \in T_{1\to 2}] C = A_{11} C,$$

where C is some upper triangular matrix whose diagonal entries are equal to 1. Hence $\text{rank}[\nabla_x \hat{m}_i(x, c^{i-1}); i \in T_{1\to 2}] = \text{rank}(A_{11}) = r_1$. The equality

$$\text{rank}[\nabla_y \hat{m}_i(y, c^{i-1}); i \in T_{2\to 1}] = r_2$$

can be shown by a similar argument. **Q.E.D.**

Let us fix some more notations. For any vector $c = (c_1, \ldots, c_r) \in \Re^r$ and $1 \le i \le n$, we let

$$\begin{aligned} S(c) &= \{\ (x,y) \in D_x \times D_y \mid m_i(x,y) = c_i,\ i = 1, \ldots, r\ \}, \\ S_x(c) &= \{\ x \in D_x \mid \hat{m}_i(x, c^{i-1}) = c_i,\ \forall i \in T_{1\to 2}\ \}, \\ S_y(c) &= \{\ y \in D_y \mid \hat{m}_i(y, c^{i-1}) = c_i,\ \forall i \in T_{2\to 1}\ \}, \\ R^r &= \{\ (m_1(x,y), \ldots, m_r(x,y)) \mid (x,y) \in D_x \times D_y\ \}. \end{aligned} \quad (3.5.10)$$

Theorem 3.5.3 *For any $c \in R^r$, we have*

$$S(c) = S_x(c) \times S_y(c). \quad (3.5.11)$$

Proof: We have, using the definition (3.5.10) and Eqs. (3.5.2)–(3.5.3),

$$\begin{aligned} S(c) &= \{\ (x,y) \in D_x \times D_y \mid \hat{m}_i(x, c^{i-1}) = c_i,\ \forall i \in T_{1\to 2}, \\ &\qquad\qquad \hat{m}_i(y, c^{i-1}) = c_i,\ \forall i \in T_{2\to 1}\ \} \\ &= S_x(c) \times S_y(c). \end{aligned}$$

Q.E.D.

We now fix some $(x^*, y^*) \in \overline{D}$ and let $c^* = m(x^*, y^*)$. Consider a map F given by

$$F_i(x, c) = \hat{m}_i(x, c^{i-1}) - c_i, \qquad \forall c \in R^r,\ x \in D_x,\ i \in T_{1\to 2}.$$

Notice that $F(x^*, c^*) = 0$. Moreover, it follows from Theorem 3.5.2 that the matrix $[\nabla_x F(x^*, c^*)]$ has full rank. It is now clear that we are in a position to apply Theorem A.1.2 in the appendix (with the correspondence $u \leftrightarrow x$, and $v \leftrightarrow c$) to conclude that there exist an open subset U_1 of $\Re^r$ containing c^*, an open subset $\hat{D}_x$ of D_x containing x^* such that $S_x(c) \bigcap \hat{D}_x$ is nonempty and connected for all $c \in U_1$. Following a symmetrical argument, we see that there exist open subsets $U_2 \subset \Re^r$ and $\hat{D}_y \subset D_y$ such that $c^* \in U_2$, $y^* \in \hat{D}_y$, and $S_y(c) \bigcap \hat{D}_y$ is nonempty and connected for all $c \in U_2$. Let $U = U_1 \bigcap U_2$. Clearly, U is nonempty since $c^* \in U$. In light of Theorem 3.5.3, we see that for all $c \in U$,

$$\begin{aligned} \hat{S}(c) &\triangleq S(c) \bigcap \left(\hat{D}_x \times \hat{D}_y\right) \\ &= \left(S_x(c) \bigcap \hat{D}_x\right) \times \left(S_y(c) \bigcap \hat{D}_y\right), \end{aligned}$$

and the set $\hat{S}(c)$ is nonempty and connected. In what follows, we let

$$\hat{S}_x(c) = S_x(c) \bigcap \hat{D}_x, \qquad \hat{S}_y(c) = S_y(c) \bigcap \hat{D}_y. \tag{3.5.12}$$

Theorem 3.5.4 *There exists some $c^* \in U$ such that $h(x, c^*)$ is a nonconstant function of x on the set $\hat{S}_x(c^*)$.*

Proof: Since we have assumed that the final result is evaluated by processor P_1, it follows that the last message $m_r(x, y)$ must have been sent by processor P_2. (Otherwise, processor P_1 would be able to evaluate $f(x, y)$ on the basis of $m_1(x, y), \ldots, m_{r-1}(x, y)$, and we would have a protocol with $r - 1$ messages, thus contradicting Eq. (3.5.1).) We prove Theorem 3.5.4 by contradiction. Suppose that there exists some function $w : U \mapsto \Re$ such that

$$h(x, c) = w(c), \quad \forall c \in U,\ \forall x \in \hat{S}_x(c), \tag{3.5.13}$$

where h is the function given by Eq. (3.5.4). We claim that w is a continuous function of c in U. In fact, let c be an arbitrary vector in U and let $\{c_i \in U; i = 1, 2, \ldots\}$

be a sequence of vectors converging to c. By Theorem A.1.2 in the appendix, we can pick a convergent sequence of vectors $\{x_i \in \hat{S}_x(c_i); i = 1, 2, \ldots\}$ such that $\lim_{i\to\infty} x_i = x$ for some $x \in \hat{D}_x$. By using Eq. (3.5.13) and the continuity of h, we see that

$$\lim_{i\to\infty} w(c_i) = \lim_{i\to\infty} h(x_i, c_i) = h(x, c),$$

which implies that w is continuous on U. Since for any $(x, y) \in m^{-1}(U) \bigcap \left(\hat{D}_x \times \hat{D}_y\right)$ we have $m(x, y) \in U$, Eq. (3.5.13) yields

$$f(x, y) = h(x, m(x, y)) = w(m(x, y)), \qquad \forall (x, y) \in \hat{D}_x \times \hat{D}_y.$$

Thus, f can be evaluated on the basis of $m(x, y)$ alone over the set $m^{-1}(U) \bigcap \left(\hat{D}_x \times \hat{D}_y\right)$ and this can be done by processor P_2 before sending the last message. Thus, Eq. (3.5.13) leads to a protocol with $r-1$ messages for computing f over $m^{-1}(U) \bigcap \left(\hat{D}_x \times \hat{D}_y\right)$. This will contradict Eq. (3.5.1) once we show that $m^{-1}(U) \bigcap \left(\hat{D}_x \times \hat{D}_y\right)$ is a nonempty open set. To this effect, we notice that $\hat{S}(c)$ is nonempty and that

$$\hat{S}(c) \subset m^{-1}(U) \bigcap \left(\hat{D}_x \times \hat{D}_y\right), \qquad \forall c \in U,$$

from which it follows that $m^{-1}(U) \bigcap \left(\hat{D}_x \times \hat{D}_y\right)$ is nonempty. Furthermore, $m^{-1}(U)$ is open since it is the inverse image of the open set U under a continuous mapping. Thus, $m^{-1}(U) \bigcap \left(\hat{D}_x \times \hat{D}_y\right)$ is open, since $\hat{D}_x \times \hat{D}_y$ is open by construction. **Q.E.D.**

3.5.2 A General Lower Bound

Let $f : D_x \times D_y \mapsto \Re$ be a continuously differentiable function, where D_x and D_y are some open subsets of $\Re^m$ and $\Re^n$, respectively. We use the notation $\nabla_x f(x, y)$ and $\nabla_y f(x, y)$ to denote the m–dimensional (respectively, n–dimensional) vector whose components are the partial derivatives of f with respect to the components of x (respectively, y). Also, for any set $S \subset D_x$, we use $[\nabla_y f(x, y); x \in S]$ to denote the subspace of $\Re^n$ spanned by the vectors $\nabla_y f(x, y)$, $x \in S$. Finally, for any set $S \subset D_y$, $[\nabla_x f(x, y); y \in S]$ is similary defined.

Assumption 3.5.1 *For any $y \in D_y$, we let*

$$S(y) = \{\ S \subset D_x \mid f(S, y) \text{ contains an open interval }\}.^5$$

(*For any $x \in D_x$, $S(x)$ is similarly defined.*)

a) *For any $y \in D_y$ and any nonempty open set $S \subset D_x$, we have $S \in \mathcal{S}(y)$.*

b) *For any $x \in D_x$ and any nonempty open set $S \subset D_y$, we have $S \in \mathcal{S}(y)$.*

c) *For some nonnegative integer n_f, we have*

$$\dim[\nabla_y f(x,y); x \in S] \geq n_f, \qquad \forall y \in D_y,\ \forall S \in \mathcal{S}(y). \tag{3.5.14}$$

d) *For some nonnegative integer m_f, we have*

$$\dim[\nabla_x f(x,y); y \in S] \geq m_f, \qquad \forall x \in D_x,\ \forall S \in \mathcal{S}(x). \tag{3.5.15}$$

Our main result is the following,

Theorem 3.5.5 *There holds*

$$C_1(f; D_x \times D_y) \geq \min\{n_f, m_f\}. \tag{3.5.16}$$

Proof: Let $r = C_1(f; D_x \times D_y)$. We first prove that it is sufficient to show the lower bound (3.5.16) under the additional assumption

$$r = \min_{\overline{D}_x, \overline{D}_y} C_1(f; \overline{D}_x \times \overline{D}_y), \tag{3.5.17}$$

where the minimum is taken over all nonempty open subsets $\overline{D}_x, \overline{D}_y$ of D_x, D_y, respectively. In fact, suppose that we have already shown that Theorem 3.5.5 is true under the assumption (3.5.17). Let us now show (3.5.16) when Eq. (3.5.17) does not hold. In this case, there exists some $r' < r$ and some open subsets $\hat{D}_x \times \hat{D}_y$ of $D_x \times D_y$ such that

$$r' = C_1(f; \hat{D}_x \times \hat{D}_y) = \min_{\overline{D}_x, \overline{D}_y} C_1(f; \hat{D}_x \times \hat{D}_y).$$

where the minimum is taken over all nonempty open subsets $\overline{D}_x$, $\overline{D}_y$ of $\hat{D}_x$, $\hat{D}_y$. Thus, Eq. (3.5.17) holds with r, D_x and D_y replaced by r', $\hat{D}_x$ and $\hat{D}_y$ respectively.

[5] The notation $f(S,y)$ stands for the set $\{f(x,y) \mid x \in S\}$. Similar notation will be used later without further comment.

Since any nonempty open subset of $\hat{D}_x$ (respectively, $\hat{D}_y$) is also a nonempty subst of D_x (respectivley, D_y), we see that Assumption 3.5.1 remains valid (with the same constants n_f, m_f) when D_x, D_y are repalced by $\hat{D}_x$, $\hat{D}_y$. Therefore, Theorem 3.5.5 applies and shows that $r > r' \geq \min\{n_f, m_f\}$, which shows that Theorem 3.5.5 holds regardless of assumption (3.5.17).

In the rest of the proof, we will assume that (3.5.17) holds. As a result, all the results of Subsection 3.5.1 apply. Let us consider a protocol that uses exactly r messages, described by (cf. Chapter 1)

$$\begin{aligned} m_i(x,y) &= \hat{m}_i(x, m_1(x,y), \ldots, m_{i-1}(x,y)), \quad \forall (x,y) \in D_x \times D_y, \text{ if } i \in T_{1\to 2}, \\ m_i(x,y) &= \hat{m}_i(y, m_1(x,y), \ldots, m_{i-1}(x,y)), \quad \forall (x,y) \in D_x \times D_y, \text{ if } i \in T_{2\to 1}, \end{aligned}$$

where each m_i and $\hat{m}_i$ is a continuously differentiable function. Without loss of generality, we assume that the final evaluation of f is performed by processor P_1. Thus, there exists some continuous function h such that

$$f(x,y) = h(x, m_1(x,y), \ldots, m_r(x,y)), \quad \forall (x,y) \in D_x \times D_y.$$

By Theorem 3.5.4 of Subsection 3.5.1, we conclude that there exist open subsets $\hat{D}_x \subset D_x$ and $\hat{D}_y \subset D_y$ and some vector $(\hat{x}, \hat{y}) \in \hat{D}_x \times \hat{D}_y$ such that the set $\hat{S}_x(\hat{c})$ (cf. Eq. (3.5.12)) is nonempty and connected, where $\hat{c} = m(\hat{x}, \hat{y})$, and $h(x, \hat{c})$ is a nonconstant function of x on $\hat{S}_x(\hat{c})$. Since h is a continuous function and the set $\hat{S}_x(\hat{c})$ is nonempty and connected, we see that $h\left(\hat{S}_x(\hat{c}), \hat{c}\right)$ must contain an open interval in $\Re$. Using the fact that $f(x,y) = h(x,\hat{c})$ for all $(x,y) \in \hat{S}_x(\hat{c}) \times \hat{S}_y(\hat{c})$, we have

$$f(\hat{S}_x(\hat{c}), y) = h\left(\hat{S}_x(\hat{c}), \hat{c}\right), \quad \forall y \in \hat{S}_y(\hat{c}).$$

Therefore, $f(\hat{S}_x(\hat{c}), y)$ contains an open interval, or equivalently, $\hat{S}_x(\hat{c}) \in S(y)$ for all $y \in \hat{S}_y(\hat{c})$ (cf. Definition 3.5.1). Let us fix some $\hat{y} \in \hat{S}_y(\hat{c})$. Then, using the definition of n_f (Eq. (3.5.14)), there exist $x^1, \ldots, x^{n_f} \in \hat{S}_x(\hat{c})$ such that $\nabla_y f(x^1, \hat{y}), \ldots, \nabla_y f(x^{n_f}, \hat{y})$ are linearly independent. Meanwhile, we observe that

$$\hat{S}_y(\hat{c}) = \{\, y \in \hat{D}_y \mid \hat{m}_i(y, \hat{c}^{i-1}) = \hat{c}_i, \ \forall i \in T_{2\to 1} \,\}$$

and that, for any fixed $x \in \hat{S}_x(\hat{c})$, $f(x,y) = h(x,\hat{c})$ is a constant function of y on the set $\hat{S}_y(\hat{c})$. Moreover, by Theorem 3.5.2, we have

$$\text{rank}[\nabla_y \hat{m}_i(y, \hat{c}^{i-1}); i \in T_{2\to 1}] = r_2, \quad \forall y \in \hat{D}_y. \tag{3.5.18}$$

Thus, we are now in a position to apply Theorem A.1.3 (with the correspondence $A \leftrightarrow \hat{S}_y(\hat{c})$, $F \leftrightarrow \{\hat{m}_i(y, \hat{c}^{i-1}) - c_i; i \in T_{2\to 1}\}$) and conclude that

$$\nabla_y f(x, \hat{y}) \in \text{span } \{\nabla_y \hat{m}_i(\hat{y}, \hat{c}^{i-1}),\ i \in T_{2\to 1}\}, \qquad \forall x \in \hat{S}_x(\hat{c}).$$

Since each $x^j \in \hat{S}_x(\hat{c})$, we see that $\nabla_y f(x^j, \hat{y})$ is in the span of the vectors $\{\nabla_y \hat{m}_i(\hat{y}, \hat{c}^{i-1})$, $i \in T_{2\to 1}\}$, for $j = 1, \ldots n_f$. Using the fact that the vectors $\nabla_y f(x^j, \hat{y})$ are linearly independent, we conclude that $r \geq r_2 \geq n_f \geq \min\{m_f, n_f\}$ which is the desired result, under the assumption that processor P_1 performs the final evaluation of f. A similar argument yields $r \geq r_1 \geq n_f \geq \min\{m_f, n_f\}$ for the case where processor P_2 performs the final evaluation of f. **Q.E.D.**

As a remark, we notice that in the preceding proof we have actually shown that $r_2 \geq n_f$ in the case where processor P_1 performs the final computation and $r_1 \geq m_f$ if processor P_2 performs the final computation. Therefore, if $C_1(f; D_x \times D_y) = \min\{m_f, n_f\}$, then either $r_1 = m_f$ and $r_2 = 0$, or, $r_1 = 0$ and $r_2 = n_f$. This means that our lower bound is tight only for those problems for which one–way communication protocols are optimal.

Corollary 3.5.1 *If $C_1(f; D_x \times D_y) = \min\{n_f, m_f\}$, then any optimal communication protocol for computing f over $D_x \times D_y$ is necessarily an one–way communication protocol.*

3.5.3 Computing a Root of a Polynomial Revisited

As an application of Theorem 3.5.5, we examine the communication complexity of computing a root of a polynomial. We shall demonstrate that in this case Abelson's result is far from being optimal.

Let $x = (x_0, \ldots, x_{n-1}) \in \Re^n$ and $y = (y_0, \ldots, y_{n-1}) \in \Re^n$; let $F(z; x, y)$ be the polynomial in the scalar variable z defined by

$$F(z; x, y) = \sum_{i=0}^{n-1} (x_i + y_i) z^i, \tag{3.5.19}$$

Processor P_1 (respectively, P_2) has access to the vector x (respectively, y) and the objective is the computation of a particular root of the polynomial of $F(z; x, y)$. In

order for the problem to be well–defined, we must specify which one of the $n-1$ roots of the polynomial is to be computed. This is accomplished as follows. We fix some $(x^*, y^*) \in \Re^{2n}$ such that one of the roots (call it z^*) of the polynomial $F(z; x^*, y^*)$ is real and simple. This root will vary continuously and will remain a real and simple root as x and y vary in some open set containing x^*, y^*. We formulate this discussion in the following result.

Lemma 3.5.1 *Suppose that z^* is a real and simple root of $F(z; x^*, y^*)$. Then, there exist open sets D_x, $D_y \subset \Re^n$ such that $(x^*, y^*) \in D_x \times D_y$ and an infinitely differentiable function $f : D_x \times D_y \mapsto \Re$ such that $f(x^*, y^*) = z^*$ and*

$$F(f(x,y); x, y) = 0, \qquad \forall (x,y) \in D_x \times D_y. \tag{3.5.20}$$

Proof: Notice that $\frac{\partial F}{\partial z}(z^*; x^*, y^*) \neq 0$, since z^* is a simple root. By the implicit function theorem ([S 65, page 41]), we see that there exists an open set D containing (x^*, y^*) and an infinitely differentiable function $g : D \mapsto \Re$ such that $g(x^*, y^*) = z^*$ and $F(g(x,y); x, y) = 0$ for all $(x,y) \in D$. Now by the continuity of $\frac{\partial F}{\partial z}(z; x, y)|_{z=g(x,y)}$ at the point (x^*, y^*), there exist open sets D_x, D_y such that $(x^*, y^*) \in D_x \times D_y \subset D$ and such that $\frac{\partial F}{\partial z}(z; x, y)|_{z=g(x,y)} \neq 0$ for all $(x,y) \in D_x \times D_y$. As a result, $g(x,y)$ is a simple root of the polynomial equation $F(z; x, y) = 0$ for all $(x,y) \in D_x \times D_y$. Let f be the restriction of g on $D_x \times D_y$. Clearly, f has all the desired properties. **Q.E.D.**

By Lemma 3.5.1, we see that $f(x,y)$ is a root of $F(z; x, y)$ and is a well–defined smooth map from $D_x \times D_y$ to $\Re$. We are interested in the communication complexity $C_1(f; D_x \times D_y)$ of computing $f(x,y)$ as (x,y) varies in the set $D_x \times D_y$. We start by pointing out that Abelson's lower bound (Theorem 3.2.1) is rather weak.

Lemma 3.5.2 *The rank of the matrix $H_{xy}(f)$, whose (i,j)-th entry is equal to $\frac{\partial^2 f}{\partial x_i \partial y_j}$, is at most 3, for any $(x,y) \in D_x \times D_y$.*

Proof: We have

$$\sum_{i=0}^{n-1} (x_i + y_i)(f(x,y))^i = 0, \qquad \forall (x,y) \in D_x \times D_y.$$

We differentiate both sides of the above equation,with respect to y_m, to obtain

$$\sum_{i=1}^{n-1} i(x_i+y_i)(f(x,y))^{i-1}.\frac{\partial f(x,y)}{\partial y_m}+(f(x,y))^m = 0, \qquad \forall (x,y) \in D_x \times D_y,\ 0 \leq m \leq n-1. \tag{3.5.21}$$

We differentiate Eq. (3.5.21) further, with respect to x_l, to obtain

$$\sum_{i=1}^{n-1} i(i-1)(x_i+y_i)(f(x,y))^{i-2}\frac{\partial f(x,y)}{\partial x_l}\frac{\partial f(x,y)}{\partial y_m} + \sum_{i=1}^{n-1} i(x_i+y_i)(f(x,y))^{i-1}\frac{\partial^2 f(x,y)}{\partial x_l \partial y_m}$$
$$+m(f(x,y))^{m-1}\frac{\partial f(x,y)}{\partial x_l} + l(f(x,y))^{l-1}\frac{\partial f(x,y)}{\partial y_m} = 0. \tag{3.5.22}$$

Since $f(x,y)$ is a simple root, it follows that $\sum_{i=1}^{n-1} i(x_i+y_i)(f(x,y))^{i-1} \neq 0$. Equation (3.5.22) shows that $\frac{\partial^2 f(x,y)}{\partial x_l \partial y_m}$ is of the form $u_1(l)v_1(m)+u_2(l)v_2(m)+u_3(l)v_3(m)$, where $u_i(l), v_i(m)$ are some real numbers depending on x, y. Therefore the rank of the matrix $H_{xy}(f)$ can be at most 3, for any point $(x,y) \in D_x \times D_y$. **Q.E.D.**

We now illustrate the power of our general results of Subsection 3.5.2, by deriving a lower bound that matches the obvious upper bound.

Theorem 3.5.6 *Let D_x, D_y be as in Lemma 3.5.1. Then, $C_1(f(x,y); D_x \times D_y) = n$.*

Proof: The upper bound $C_1(f; D_x \times D_y) \leq n$ is obvious, so we concentrate on the proof of the lower bound. To this effect, we will employ Theorem 3.5.5 and it suffices to verify that Assumption 3.5.1 holds with $n_f = m_f = n$. Since the roots of a polynomial equation cannot remain constant when the coefficients vary over an open set, it follows that the continuous function $f(x,y)$ given by Lemma 3.5.1 satisfies parts (a) and (b) of Assumption 3.5.1. Now we fix some $y \in D_y$ and some $S \in \mathcal{S}$, that is, $S \subset D_x$ and $f(S,y)$ contains an open interval. Let $c_1, \ldots, c_n$ be some distinct real numbers in $f(S,y)$ and $x^1, \ldots, x^n \in S$ such that

$$f(x^i, y) = c_i, \qquad i = 1, \ldots, n. \tag{3.5.23}$$

Let x_j^i be the j-th coordinate of x^i. Using Eq. (3.5.21), we see that

$$a_i \nabla_y f(x^i, y) = -\begin{bmatrix} 1 \\ c_i \\ \vdots \\ c_i^{n-1} \end{bmatrix}, \tag{3.5.24}$$

where $a_i = \sum_{j=1}^{n-1} j(x_j^i + y_j)c_i^{j-1}$. If we form a matrix whose colums are the vectors $(1, c_i, \ldots, c_i^{n-1})$, $i = 1, \ldots, n$, this matrix is a Vandermonde matrix and is nonsingular, because the values $c_1, \ldots, c_n$ are chosen to be distinct. Then, Eq. (3.5.24) implies that the vectors $\nabla_y f(x^i, y)$, $i = 1, \ldots, n$, are linearly independent. This proves that $n_f = n$. The proof that $m_f = n$ is similar. **Q.E.D.**

As a remark, we point out that Theorem 3.5.6 is in some sense the strongest result possible. The only assumptions we used in showing Theorem 3.5.6 are that (a) the message functions are continuously differentiable; (b) the final evaluation function is a continuous function; (c) the protocol computes a root of a polynomial on some open set. As discussed in Chapter 1, assumption (a) is necessary since its removal could lead to unreasonable conclusions. Assumption (b) is basic and natural since the function to be computed, i.e., a particular real simple root of some polynomial, is continuous, while assumption (c) is minimal. Finally, we note that no truly two–way communication protocol can be optimal. In other words, if each processor transmits at least one message, then at least $n + 1$ messages have to be exchanged. This is a simple consequence of Corollary 3.5.1 of Subsection 3.5.2.

3.5.4 Comparison With Abelson's Bound

In the previous subsection, we have seen that Theorem 3.5.5 can yield a much better bound than Abelson's result (Theorem 3.2.1). However, it is not true, as we shall see next, that Theorem 3.5.5 always provides a stronger lower bound. The reason is, loosely speaking, that our result only places a constraint on the minimum number of messages that has to to be sent by a single processor, while Abelson's result is a bound on the total number of messages sent by both processors. As pointed out at the end of Subsection 3.5.2, any two–way communication protocol that attains the lower bound in Theorem 3.5.5 is necessarily an one–way protocol. Notice that our result makes use of information about the first order derivatives of function f. This is in contrast to Abelson's result which uses only the second order derivatives of f. In what follows, we provide an example where Abelson's bound is more effective than our bound.

Example: Let $f(x, y) = x^T Q y$, where Q is some $m \times n$ matrix and $x \in \Re^m$ and

$y \in \Re^n$. By Theorem 3.2.1, we see that $C_2(f; \Re^m \times \Re^n) \geq \text{rank}(Q)$. Using the singular value decomposition of Q (see [GV 83]), one can construct a protocol that uses exactly rank(Q) messages (see Theorem 3.2.2). Therefore, we conclude that $C_2(f; \Re^m \times \Re^n) = \text{rank}(Q)$. Next we apply Theorem 3.5.5 to f. To this effect, we need to find out of the values m_f and n_f.

Suppose that rank$(Q) = r > 0$. Let D_x, D_y be some connected open subsets of $\Re^m$ and $\Re^n$ respectively. We assume that $0 \notin D_x$ and $0 \notin D_y$ in which case $f(x,y)$ is nonconstant as x or y vary in an open subset of D_x or D_y, respectivley. Thus, parts (a) and (b) of Assumption 3.5.1 are satisfied. We now show that Assumption 3.5.1 can only hold with $\min\{m_f, n_f\} \leq 2$. By the singular value decomposition, there exist two linearly independent families of vectors $u_1, \ldots, u_r$ in $\Re^m$ and $v_1, \ldots, v_r$ in $\Re^n$, such that

$$Q = u_1 v_1^T + u_2 v_2^T + \cdots + u_r v_r^T. \tag{3.5.25}$$

It follows that $x^T Q y = \sum_1^r (u_i^T x)(v_i^T y)$. Since $r > 0$, there exists some point $(x_0, y_0) \in D_x \times D_y$ such that $x_0^T Q y_0 \neq 0$. Hence, we can, without loss of generality, assume that $(u_r^T x_0)(v_r^T y_0) \neq 0$. Let $S = \{x \in D_x \mid u_i^T x = u_i^T x_0,\ 1 \leq i \leq r-1\}$. Clearly, S is nonempty since $x_0 \in S$. We claim that if $r > 1$ then $f(S, y_0)$ contains an open interval. In fact, equation (3.5.25) shows that

$$\begin{aligned} x^T Q y_0 &= \sum_{i=1}^{r} (u_i^T x)(v_i^T y_0) \\ &= \sum_{i=1}^{r-1} (u_i^T x_0)(v_i^T y_0) + (u_r^T x)(v_r^T y_0), \qquad \forall x \in S. \end{aligned} \tag{3.5.26}$$

Since u_r is linearly independent from $u_1, \ldots, u_{r-1}$, we see that $u_r^T x$ is a nonconstant function of x on S. Using the fact that $v_r^T y_0 \neq 0$ and Eq. (3.5.26), we see that $x^T Q y_0$ is also a nonconstant function of x on the set S. Note that S is connected because D_x is assumed to be connected. It follows that $f(S, y_0)$ contains an open interval. To see that $n_f \leq 2$, we notice that

$$\nabla_y f(x, y_0) = \sum_{i=1}^{r-1} (u_i^T x_0) v_i + (u_r^T x) v_r, \qquad \forall x \in S.$$

Hence, $\dim[\nabla_y f(x, y_0); x \in S] \leq 2$. Thus, Assumption 3.5.1 can only hold with $n_f \leq 2$. The relation $m_f \leq 2$ can be established in a symetrical fashion. As a result, we have shown that $\min\{m_f, n_f\} \leq 2$.

Thus, for the problem $f(x,y) = x^T Qy$, Theorem 3.5.5 provides a lower bound of at most 2 as opposed to the lower bound of rank(Q) provided by Abelson's result. Hence, Theorem 3.5.5 can be quite far from optimal in general. Furthermore, the above example and the results of Subsection 3.5.2 illustrate that Theorems 3.2.1 and 3.5.5 are incomparable.

3.6 Discussion and Possible Extensions

In this chapter, we have studied the two–way continuous communication complexity for several classes of communication protocols. We have used two different approaches in deriving new lower bounds for these classes of communication protocols. The first approach is to combine Abelson's result with Hilbert's Nullstellensatz so as to exploit the algebraic structures present in the problems under consideration. The results obtained using this approach are included in Section 3.2. To illustrate the power of these results, we considered an important application in which $f(x,y)$ (the function to be computed) is defined to be a particular entry of $[(x+y)]^{-1}$, where x and y are matrices of size $n \times n$, and obtain a lower bound of $n^2 - 1$, as opposed to the $\Omega(n)$ lower bound obtained by applying Abelson's result. Our second approach is based on the observation that if we restrict the messages sent by processor P_2 at some fixed values, then $f(x,y)$ must be a function of x only and some information can be inferred about y from the values of $f(x,y)$. Thus, any protocol that computes f has to exchange enough messages to ensure that certain amount of information about y is sent to processor P_2. Our second approach is analytical in nature and the result (cf. Subsection 3.5.2) is very different from that of Abelson's because our lower bound has to do only with the first order derivatives of f, as opposed to the second order derivatives used in the Abelson's result. We then applied our new lower bound to the problem in which f is given as a root z of the polynomial $\sum_{i=0}^{n-1}(x_i + y_i)z^i$, and obtained a lower bound of n. This is in contrast to the $\Omega(1)$ lower bound obtained by applying Abelson's result to this problem. We also pointed out a situation in which Abelson's bound becomes far superior to ours. Thus, our result and Abelson's result are in general incomparable. Finally, we studied the two–way continuous communication complexity for

computing multiple functions. We demonstrated how, in some situations, certain techniques from combinatorial optimization can be applied to obtain an optimal communication protocol.

Several important questions concerning two–way continuous communication complexity remain unresolved. First of all, we have not been able to provide a complete characterization of the two–way communication complexity of computing f. We believe that such charaterization must take into account the derivatives (of f) of all orders. One can view the results of Section 3.5 as a step in this direction. Second, we believe that the results of Section 3.2 have not exploited the algebraic structures completely and these results can be strengthened by using more powerful algebraic tools from modern mathematics. In particular, we conjecture that $C_{prat}(f; D) = n^2$ for any open subset $D \subset D_f$, where $f = [(x + y)]_{11}^{-1}$ and $D_f = \{(x, y) \in \Re^{2n} | \det(x + y) \neq 0\}$. In addition, the question whether $C_{ppoly}(fg; D) \geq C_{ppoly}(f; D)$, where f, g are some nonzero polynomials and D is an open subset of $\Re^{m+n}$, is also important. Besides, it would be interesting to find more applications of the results in Section 3.2. Finally, for the problem of computing multiple functions, it remains to be seen if the combinatorial issues, such as the ones described in Section 3.4, are indeed computationally intractable. We believe that more tools from combinatorial optimization should be used to study this problem.

Chapter 4

Two–Way Discrete Communication Model

In previous chapters, we have studied communication complexity under communication models in which the messages are assumed to be real (or complex) functions. In this chapter, we focus on discrete communication protocols in which the messages are binary strings rather than real–valued functions, and develop lower bounds for several problems. In particular, we show (Section 4.1) that Abelson's result on two–way continuous communication complexity remains valid for discrete communication protocols for computing a Boolean function. In Section 4.2, we consider a situation where two processors P_1, P_2 are to minimize the sum of two convex functions f_1, f_2, defined on a common bounded domain $\mathcal{D} \subset \Re^n$, under the assumption that processor P_i has access only to f_i $(i = 1, 2)$. The minimization is performed only approximately and the information exchange is stopped as soon as a vector $x^* \in \mathcal{D}$ is found such that $f_1(x^*) + f_2(x^*) \leq \min_{x \in \mathcal{D}}[f_1(x) + f_2(x)] + \epsilon$, where $\epsilon > 0$ is some prescribed accuracy. Under some mild assumptions on f_i $(i = 1, 2)$, we prove a lower bound of $\Omega(\log \epsilon)$ and an upper bound of $O(\log^2 \epsilon)$ on the number of bits that have to be exchanged. When the domain of f_i $(i = 1, 2)$ is $[0, 1]$, we show that $O(\log \epsilon)$ bits of information exchange is sufficient. In the case where f_i $(i = 1, 2)$ are strongly convex functions, the lower bound $\Omega(\log \epsilon)$ is also shown to be tight. In Section 4.3, we prove that two–way communication protocols can sometimes provide exponential savings in the number of bits to be exchanged. Our proof makes use of the results of Section 4.2 and a result of R. Dudley ([D 74]) on how to approximate a bounded convex body by polygons. In the final section (Section 4.4), we discuss

the results obtained in this chapter and mention several important open problems.

4.1 Distributed Boolean Computation

In this section, we consider a situation where two processors P_1, P_2 are to compute some Boolean function $f(x,y)$ ($x \in \{0,1\}^m$, $y \in \{0,1\}^n$), assuming that processor P_1 (respectively, P_2) has access to the variable x (respectively, y) only. It is assumed that the messages to be exchanged between the two processors are binary digits. We wish to determine a protocol for computing f which uses as few binary messages as possible. In what follows, we show that Abelson's lower bound (see Theorem 3.2.1 of Chapter 3) remain valid for such discrete Boolean computation.

Let $f : D_x \times D_y \mapsto \Omega$ be some function, where $x \in D_x$, $y \in D_y$. In the case where D_x, D_y are open subsets of $\Re^m$ and $\Re^n$ respectively and $\Omega = \Re$ (the real line), Abelson has shown, under the assumption that f is twice continuously differentiable, that (see Theorem 3.2.1) $C_2(f; D_x \times D_y) \geq \text{rank} H_{xy}(f)|_p$, for all $p \in D_x \times D_y$, where $H_{xy}(f)|_p$ denotes the $m \times n$ matrix whose (i,j)-th entry is given by $\frac{\partial^2 f}{\partial x_i \partial y_j}(p)$. In the case where f is a Boolean function, the sets D_x, D_y and Ω are, respectively, $\{0,1\}^m$, $\{0,1\}^n$ and $\{0,1\}$. Thus, f does not seem to satisfy the differentiability requirement that is needed in order to apply Abelson's result. But, as we show next, after $\{0,1\}^m$ and $\{0,1\}^n$ are given certain algebraic structures, the Boolean function f will acquire some analytic properties which will make Abelson's bound applicable to discrete communication protocols. In fact, we let $\mathcal{F} = \{0,1; \oplus, \otimes\}$ be the field over $\{0,1\}$ for which $\oplus$ and $\otimes$ are defined as the usual modular 2 addition and multiplication operators. Note that each polynomial map from $\mathcal{F}^n$ to $\mathcal{F}$ is of the form $f = \sum_{i_1,\ldots,i_n} a_{i_1 \cdots i_n} x_1^{i_1} \cdots x_n^{i_n}$, where each coefficient $a_{i_1 \cdots i_n}$ is some element of $\mathcal{F}$ and each x_{i_k} is a variable in $\mathcal{F}$. It is easily seen that each polynomial map from $\mathcal{F}^n$ to $\mathcal{F}$ can be viewed as a Boolean function defined on $\{0,1\}^n$. The fact that the converse is also true is the subject of next lemma.

Lemma 4.1.1 *Let Σ_1 be the set of Boolean functions defined on $\{0,1\}^n$ and let*

$$\Sigma_2 = \{ f \mid f : \mathcal{F}^n \mapsto \mathcal{F} \text{ is a polynomial map. } \}$$

Then, there exists an one–to–one mapping $\phi : \Sigma_1 \mapsto \Sigma_2$ *such that* $\phi(\Sigma_1) = \Sigma_2$.

Proof: Notice that every $f \in \Sigma_2$ can be viewed as a Boolean function $\hat{f} : \{0,1\}^n \mapsto \{0,1\}$ in a very natural way: for each $(x_1, \ldots, x_n) \in \{0,1\}^n$, we can think of $(x_1, \ldots x_n)$ as an element of $\mathcal{F}^n$, and define $\hat{f}(x_1, \ldots, x_n)$ to be $f(x_1, \ldots, x_n)$, since $f(x_1, \ldots, x_n)$ can be thought of as an element of $\{0,1\}$. Thus, we can define a map $\psi : \Sigma_2 \mapsto \Sigma_1$ by letting $\psi(f) = \hat{f}$. The rest of the proof consists of proving ψ is an one–to–one and onto mapping.

Let $f = \sum_{i_1,\ldots,i_n} a_{i_1 \cdots i_n} x_1^{i_1} \cdots x_n^{i_n}$ be a polynomial map in Σ_2. Since $x_i^2 = x_i$ for all $x_i \in \mathcal{F}$, it follows that f can be written in the form $\sum_{I \subset \{1,\ldots,n\}} a_I \prod_{i \in I} x_i$, where a_I is some element of $\mathcal{F}$. Now let $g = \sum_{I \subset \{1,\ldots,n\}} b_I \prod_{i \in I} x_i$ be another polynomial map in Σ_2. We claim that $\psi(f) = \psi(g)$ if and only if $a_I = b_I$ for all $I \subset \{1, \ldots, n\}$. In fact, let us consider the polynomial map $f - g$. If $a_I \neq b_I$ for some I, then the corresponding coefficient of $f - g$ is nonzero. Now let I^* be a subset of the smallest cardinality such that $a_{I^*} \neq b_{I^*}$, and let

$$x_i^* = \begin{cases} 1, & \text{if } i \in I^*, \\ 0, & \text{otherwise.} \end{cases}$$

It is then easily seen that $(f - g)(x^*) = 1$, which implies that $\psi(f) \neq \psi(g)$. This proves that ψ is an one–to–one mapping. To show that ψ is also an onto mapping, let us now count the total number of polynomial maps in Σ_2. Since $f = g$ $(f, g \in \Sigma_2)$ iff $a_I = b_I$ for all I, we only need to count the total number of combinations of different choices of a_I. Since there are exactly 2^n distinct subsets I of $\{1, \ldots n\}$ and for every such subset I a_I can be either 0 or 1, it follows that the total number of polynomial maps in Σ_2 is 2^{2^n}, which is exactly equal to the number of different Boolean functions in Σ_1. This implies that ψ is an onto mapping. Now we take $\phi = \psi^{-1}$ and the proof is complete. **Q.E.D.**

Our result is the following:

Theorem 4.1.1 *Let* $f : \{0,1\}^m \times \{0,1\}^n \mapsto \{0,1\}$ *be some Boolean function, represented as a polynomial. Then,*

$$C(f; \{0,1\}^m \times \{0,1\}^n) \geq \text{rank } H_{xy}(f)|_p, \qquad p \in \{0,1\}^m \times \{0,1\}^n,$$

where the rank is evaluated over the field $\mathcal{F} = (\{0,1\}; \oplus, \otimes)$.

Proof: Let π be a protocol for computing f, described by

$$m_i(x,y) = \hat{m}_i(x, m_1(x,y), \ldots, m_{i-1}(x,y)), \qquad i \in T_{1\to 2} \tag{4.1.1}$$

and

$$m_i(x,y) = \hat{m}_i(y, m_1(x,y), \ldots, m_{i-1}(x,y)), \qquad i \in T_{2\to 1}, \tag{4.1.2}$$

where $T_{1\to 2}$ (respetively, $T_{2\to 1}$) is the set of indices i for which i-th message is sent by processor P_1 (respectively, P_2), and each $\hat{m}_i$ is a Boolean function defined on $\{0,1\}^m \times \{0,1\}^{i-1}$. Without loss of generality, let us assume that processor P_1 does the final computation. Then, there is exists a Boolean function $h : \{0,1\}^m \times \{0,1\}^{r(\pi)} \mapsto \{0,1\}$ such that

$$f(x,y) = h(x, m_1(x,y), \ldots, m_{r(\pi)}(x,y)), \qquad \forall (x,y) \in \{0,1\}^m \times \{0,1\}^n. \tag{4.1.3}$$

By Lemma 4.1.1, the Boolean functions $\hat{m}_i$, m_i $(i = 1, \ldots r(\pi))$ and f, h can be viewed as polynomials over the field $\mathcal{F}$, and as a result, we can think of Eqs. (4.1.1)–(4.1.3) as relations about these polynomial maps. (With a slight abuse of notation, we will, in the rest of this proof, use f itself to denote the polynomial corresponding to a Boolean function f.) Thus, it follows that we can take derivatives of both sides of Eqs. (4.1.1)–(4.1.3), and, after some algebraic manipulations, we obtain

$$\frac{\partial^2 f}{\partial x_i \partial y_j} = \sum_{k=1}^{r(\pi)} \Gamma_{ik} \Lambda_{kj}, \qquad (x,y) \in \mathcal{F}^m \times \mathcal{F}^n, \tag{4.1.4}$$

where Γ_{ik} is some expression which depends on i and k, and Λ_{kj} depends on j and k. (The proof of Eq. (4.1.4) can be copied verbatim from [A 80] where an analogous equation was shown through some purely algebraic manipulation.) Thus, the matrix $H_{xy}(f)$ can be written as the product of the $m \times r(\pi)$ matrix Γ and the $r(\pi) \times n$ matrix Λ. This implies that the rank of $H_{xy}(f)|_p$ can be at most $r(\pi)$, for all $p \in \mathcal{F}^m \times \mathcal{F}^n$. **Q.E.D.**

We close this section by considering the following example:

Example: Let $f : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}$ be a Boolean function defined as

$$f(x,y) = \begin{cases} 1 & \exists \text{ odd number of } i \text{ such that } x_i = y_i = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Thus, $f(x,y)$ is the parity of $x \oplus y$, which, when represented as a polynomial on $\mathcal{F}^n \times \mathcal{F}^n$, is the same as the inner product $\sum_{i=1}^n x_i y_i$. Therefore, $H_{xy}(f)|_p = I$, for all $p \in \mathcal{F}^n \times \mathcal{F}^n$. In light of Theorem 4.1.1, this implies that $C(f; \{0,1\}^n \times \{0,1\}^n) = n$.

4.2 Distributed Convex Optimization

In this section, we consider a situation where each one of two processors has access to a different convex function f_i, $i = 1,2$, defined on a common bounded convex domain. The processors are to exchange a number of binary messages, according to some protocol, until they find a point in the domain at which $f_1 + f_2$ is minimized, within some prespecified accuracy ϵ. Our objective is to determine protocols under which the number of exchanged messages is minimized.

Formally, let Θ be a set of convex functions defined on the n–dimensional bounded domain $[0,1]^n$. (Typically, Θ will be defined by imposing certain smoothness conditions on its elements.) Consider a functional $\mathcal{F} : \Theta \times \Theta \mapsto \Re$ defined by $\mathcal{F}(f_1, f_2) = \inf_{x \in [0,1]^n} (f_1(x) + f_2(x))$. Given any $\epsilon > 0$, and $f \in \Theta$, let $I(f;\epsilon)$ be the set of all $x \in [0,1]^n$ such that $f(x) \leq f(y) + \epsilon$, $\forall y \in [0,1]^n$. Thus, for each $x \in I(f_1 + f_2; \epsilon)$, $f_1(x) + f_2(x)$ is an ϵ–approximation of $\mathcal{F}(f_1, f_2)$.

Let there be two processors, denoted by P_1 and P_2, each of which is given a different convex function $f_i \in \Theta$, $i = 1,2$. Processors P_1 and P_2 are to approximately compute the functional $\mathcal{F}(f_1 + f_2)$. They start exchanging binary messages, according to some protocol π, until processor P_1 determines an element of $I(f_1 + f_2; \epsilon)$. Let $C(f_1 + f_2; \epsilon, \pi)$ be the total number of messages that are exchanged; this is a function of the particular protocol being employed and we are looking for an optimal one. More precisely, let

$$C(\mathcal{F}; \Theta, \epsilon, \pi) = \sup_{f_1, f_2 \in \Theta} C(f_1 + f_2; \epsilon, \pi) \tag{4.2.1}$$

be the communication requirement (in the worst case) of the particular protocol and let

$$C(\mathcal{F}; \Theta, \epsilon) = \inf_{\pi \in \Pi(\epsilon)} C(\mathcal{F}; \Theta, \epsilon, \pi) \tag{4.2.2}$$

be the communication requirement under an optimal protocol, where $\Pi(\epsilon)$ is the class of all protocols which work properly (see Subsection 1.1.2 of Chapter 1), for a particular choice of ϵ. The quantity $C(\mathcal{F};\Theta,\epsilon)$ may be called the ϵ- *communication complexity* of the above defined problem of distributed, approximate, convex optimization.

The communication complexity of the approximate solution of problems of continuous variables has not been studied before, to the best of our knowledge. However, there exists a large amount of theory ([TW 80], [TWW 83]) on the information requirements for solving (approximately) certain problems such as nonlinear optimization, and numerical integration of differential equations ("information based complexity"). Here one raises questions such as: how many gradient evaluations are required for an algorithm to find a point which minimizes a convex function within some prespecified accuracy ϵ? We can see that, in this type of research, information flows one–way: from a "memory unit" (which knows the function being minimized) to the processor and this is what makes it different from ours.

The rest of this section is organized as follows. In Subsection 4.2.1 we establish some straightforward lower bounds such as $C(\mathcal{F};\Theta,\epsilon) \geq \Omega(n\log(1/\epsilon))$. In Subsection 4.2.2 we show that the naive distributed version of ellipsoid–type algorithms leads to protocols with $O(n^2\log(1/\epsilon)(\log n+\log(1/\epsilon))$ communication requirements and we show that this upper bound cannot be improved substantially within a restricted class of protocols. In Subsections 4.2.3 and 4.2.4 we partially close the gap between the above mentioned upper and lower bounds by presenting protocols with $O(\log(1/\epsilon))$ communication requirements for the case $n = 1$ (Subsection 4.2.3) and with $O(n(\log n + \log(1/\epsilon))$ communication requirements for the case of general n (Subsection 4.2.4), under certain regularity assumptions on the elements of Θ. (The results of these two subsections are mainly due to John Tsitsiklis.)

4.2.1 Lower Bound on $C(\mathcal{F};\Theta,\epsilon)$

Before we prove any lower bounds we start with a fairly trivial Lemma whose proof is omitted.

Lemma 4.2.1 *If* $\Theta \subset \Delta$ *then* $C(\mathcal{F};\Theta,\epsilon) \leq C(\mathcal{F};\Delta,\epsilon)$.

Let Θ_Q be the set of quadratic functions of the form $f(x) = ||x - x^*||^2$, with $x^* \in [0,1]^n$ and where $||\cdot||$ is the Euclidean norm. Also, let Θ_M be the set of functions of the form $f(x) = \max_{i=1,...,n} |x - x_i^*|$, where $|x_i^*| \leq 1$, $\forall i$.

Proposition 4.2.1 *(i)* $C(\mathcal{F};\Theta_Q,\epsilon) \geq \Omega(n(\log n + \log(1/\epsilon)))$;
(ii) $C(\mathcal{F};\Theta_M,\epsilon) \geq \Omega(n\log(1/\epsilon))$.

Proof: (i) Consider a protocol $\pi \in \Pi(\epsilon)$ with termination time T described by

$$m_{1,t}(f_1,f_2) = \hat{m}_{1,t}(x, m_{2,0}(f_1,f_2), ..., m_{2,t-1}(f_1,f_2)), \qquad t = 1,\ldots,T, \qquad (4.2.3)$$

and

$$m_{2,t}(f_1,f_2) = \hat{m}_{2,t}(y, m_{1,0}(f_1,f_2), ..., m_{1,t-1}(f_1,f_2)), \qquad t = 1,\ldots,T, \qquad (4.2.4)$$

where $\hat{m}_{1,t} : \Theta \times \{0,1\}^t \mapsto \{0,1\}$ and $\hat{m}_{2,t} : \Theta \times \{0,1\}^t \mapsto \{0,1\}$ are the message functions sent at time t by processor P_1 and P_2 respectively. Moreover, there exists a final evaluation function $h : \Theta \times \{0,1\}^T \mapsto \Re$ such that

$$|\mathcal{F}(f_1,f_2) - h(f_1, m_{2,0}(f_1,f_2), ..., m_{2,T-1}(f_1,f_2))| \leq \epsilon, \qquad \forall (f_1,f_2) \in \Theta \times \Theta. \quad (4.2.5)$$

Let us study the operation of π for the special case where $f_1 = 0$.

Suppose that S is the range of the function h corresponding to that protocol (see equation 4.2.5), when $f_1 = 0$. Given that the minimum of f_2 may be anywhere in $[0,1]^n$, S must contain points which come within $\epsilon^{1/2}$, in Euclidean distance, from every point in $[0,1]^n$. Now, one needs at least $(\frac{An}{\epsilon^{1/2}})^{Bn}$ Euclidean balls of radius $\epsilon^{1/2}$ to cover $[0,1]^n$, where A and B are some absolute constants. (This follows by simply taking into acount the volume of a ball in n–dimensional space.) Therefore, the cardinality of S is at least $(An/\epsilon^{1/2})^{Bn}$. Given that the cardinality of the range of a function is no larger than the cardinality of its domain, it follows that the cardinality of S is no larger than 2^T. Therefore, $T \geq O(n(\log n + \log(1/\epsilon))$, which proves part (i).

(ii) The proof is almost identical to that of part (i) and is therefore omitted. The only difference is that now $[0,1]^n$ is covered by balls in the supremum norm and $O((1/\epsilon)^n)$ such balls are necessary and sufficient. **Q.E.D.**

Using Lemma 4.2.1, we conclude that $C(\mathcal{F}; \Theta_{SC,M,L}, \epsilon) \geq O(n(\log n + \log(1/\epsilon))$, where $\Theta_{SC,M,L}$ is the set of all continuously differentiable convex functions f with the property

$$L||x-y||^2 \leq \langle f'(x) - f'(y) | x-y \rangle \leq ML||x-y||^2,$$

and such that the norm of their gradient is bounded by $n^{1/2}$.

We also conclude that $C(\mathcal{F}; \Theta_L, \epsilon) \geq O(n \log(1/\epsilon)$, where Θ_L is the set of differentiable convex functions which are bounded by $1/2$ and satisfy

$$|f(x) - f(y)| \leq \frac{1}{2} \max_i |x_i - y_i|, \quad \forall x, y.$$

In the proof of Proposition 4.2.1 we made use of the assumption that the final result is always obtained by processor P_1. Nevertheless, at the cost of minor complications of the proof, the same lower bound may be obtained even if we enlarge the class of allowed protocols so that the processor who computes the final result is not prespecified.

4.2.2 Naive Upper Bounds

We consider here a straightforward distributed version of the method of the centers of gravity (MCG), which has been shown in [NY 83] to be an optimal algorithm for the single–processor case, in the sense that it requires a minimal number of gradient evaluations. This method is a generalization (and an ancestor) of the well–known ellipsoid algorithm for linear programming [PS 83]. We start by describing the uniprocessor version of this method and then analyze the communication requirements of a distributed implementation.

The MCG Algorithm([NY 83, page 62]):

Let f be a convex function to be minimized with accuracy ϵ. Let $G_0 = [0,1]^n$ and let x_0 be its center of gravity. At the beginning of the k-th stage of the computation, we assume that we are given a convex set $G_{k-1} \subset [0,1]^n$ and its center of gravity x_k. Let z_k be a scalar and let y_k be a vector in R^n with the following properties:

(i) $z_k + \langle y_k, x - x_k \rangle \leq f(x)$, $\forall x \in [0,1]^n$;
(ii) $z_k \geq f(x_k) - (\epsilon/2)$.

(Notice that if $\epsilon/2$ was absent in condition 2, then we would have $z_k = f(x_k)$ and y_k would be a subgradient of f at x_k. The presence of the $\epsilon/2$ term implies that these relations only need to hold approximately.)

Let $a_k = \min_{j \leq k}\{z_j\}$ and let $G_k = \{x \in G_{k-1} : \langle y_k, x - x_k \rangle + z_k \geq a_k\}$. The algorithm terminates when the Lebesgue volume of G_k becomes smaller than $(\epsilon/2)^n$ and returns a point x_j which has the smallest value of z_j encountered so far.

The following facts are quoted from [NY 83]:

(a) The volume of G_k is no larger than α^k, where α is an absolute constant, smaller than one and independent of the dimension n. Thus a total of $n\frac{\log(2/\epsilon)}{\log(1/\alpha)} = O(n\log(1/\epsilon))$ stages are sufficient.

(b) The result x_j of the algorithm satisfies $f(x_j) \leq \inf_{x \in [0,1]^n} f(x) + \epsilon V(f)$, where $V(f) = \sup_{x \in [0,1]^n} f(x) - \inf_{x \in [0,1]^n} f(x)$.

Notice that $V(f) \leq 1$, for $f = f_1 + f_2$, $f_1, f_2 \in \Theta_L$ so that the algorithm indeed produces a result belonging to $I(f;\epsilon)$.

We now consider a distributed implementation of this algorithm. The distributed protocol will consist of stages corresponding to the stages of the MCG algorithm. At the beginning of the k-th stage, both processors know the current convex set G_{k-1} and are therefore able to compute its center of gravity x_k. The i-th processor evaluates $f_i(x_k)$ and transmits the binary representation of a message $b(i,k)$ satisfying $b(i,k) \in [f_i(x_k) - (\epsilon/4), f_i(x_k) - (\epsilon/8)]$. Clearly, $b(i,k)$ may be chosen so that its binary representation has at most $O(\log(1/\epsilon))$ bits. Also each processor evaluates a subgradient $g_{i,k}$ of its function f_i, at x_k (with components $g_{i,k,j}$, $j = 1,...,n$) and transmits the binary representation of messages $c(i,k,j)$ satisfying $|g_{i,k,j} - c(i,k,j)| \leq \epsilon/(16n)$. Clearly the $c(i,k,j)$'s may be chosen so that

they can be all transmitted using $O(n\log(n/\epsilon)) = O(n\log n + n\log(1/\epsilon))$ bits.

Next, each processor lets $z_k = b(1,k) + b(2,k)$ and lets y_k be the vector with components $c(1,k,j) + c(2,k,j)$. It then follows by some simple algebra that z_k and y_k satisfy the specifications of the MCG algorithm. Finally, each processor determines G_k, and its center of gravity x_{k+1} and the algorithm proceeds to its next stage.

We now combine our estimates of the number of stages of the MCG algorithm and of the communication requirements per stage to conclude the following.

Proposition 4.2.2 $C(\Theta_L;\epsilon) \leq O(n^2\log(1/\epsilon)(\log n + \log(1/\epsilon))$. *This bound is attained by the distributed version of the MCG algorithm.*

The upper bound of Proposition 4.2.2 is quite far from the lower bound of Proposition 4.2.1. We show next that within a certain class of protocols this upper bound cannot be substantially improved.

We consider protocols which consist of stages. At the k-th stage there is a current point $x_k \in [0,1]^n$ known by both processors. Then, the processors transmit to each other approximate values of f_i and of a subgradient of f_i, all evaluated at x_k. Using the values of these messages, together with any past common information, they determine the next point x_{k+1}, according to some commonly known rule, and so on. We place one additional restriction: when a processor transmits an approximate value of $f_i(x_k)$ it does so by transmitting a sequence of bits of the binary representation of $f_i(x_k)$ starting from the most significant one and continuing with consecutive less significant bits. (So, for example, a processor is not allowed to transmit the first and the third most significant bits of $f_i(x_k)$, without transmitting the second most significant bit.) The same assumption is made concerning the components of the subgradient of f_i. Finally, we require that the same number of bits of $f_i(x_k)$ and of each component of the subgradient of f_i gets transmitted.

The above restrictions turn out to be quite severe.

Proposition 4.2.3 *There exists a constant A such that for any protocol $\pi \in \Pi(\epsilon)$ satisfying the above restrictions, there exist $f_1, f_2 \in \Theta_L$ such that $C(f_1,f_2;\epsilon,\pi) \geq$*

$An^2 \log^2(1/\epsilon)$. This is true, even if we restrict f_1 to be the zero function.

Proof: Using Lemma 4.2.1, it is sufficient to prove the result under the restriction that $f_1 = 0$ and under the restriction that f_2 is differentiable and bounded, together with every component of its derivative, by $\sqrt{\epsilon}$. Using the results of [NY 83], for processor P_1 to determine a point which is optimal within ϵ, it must acquire non-trivial information on the values and the derivatives of f_2 for at least $An \log(1/\sqrt{\epsilon})$ different points. Notice that the $O(\log(\sqrt{\epsilon}))$ most significant bits of f_2 and each component of its derivative, evaluated at any point, are always zero. Thus, for processor P_1 to obtain nontrivial information at a certain point at least $O(n \log(1/\sqrt{\epsilon}))$ bits have to be transmitted. This leads to a total communication requirement of $O(n^2 \log^2(1/\sqrt{\epsilon})) = O(n^2 \log^2(1/\epsilon))$ bits, which proves the result. **Q.E.D.**

If we relax the requirement that the same number of bits is transmitted for each component of the subgradient, at each stage, then the same proof yields the lower bound $C(f_1, f_2; \epsilon, \pi) \geq An \log^2(1/\epsilon)$.

4.2.3 An Optimal Algorithm for the One–Dimensional Case

We prove here a result which closes the gap between upper and lower bounds for the one–dimensional case. The proof consists of the construction of an optimal protocol.

Proposition 4.2.4 *If $n = 1$ then $C(\mathcal{F}; \Theta_L, \epsilon) \leq O(\log \frac{1}{\epsilon})$.*

Proof: The protocol consists of consecutive stages. At the beginning of the k-th stage, both processors have knowledge of four numbers a_k, b_k, c_k and d_k with the following properties:

(i) The interval $[a_k, b_k]$ contains a point x^* which minimizes $f_1 + f_2$.

(ii) The derivative of f_1 at any minimizer of $f_1 + f_2$ and the derivative of f_1 and of $-f_2$ at $(a_k + b_k)/2$ belong to the interval $[c_k, d_k]$. (Notice that the derivative of each f_i has to be constant on the set of minimizers of $f_1 + f_2$.)

At the first stage of the algorithm we start with $a_1 = 0$, $b_1 = 1$, $c_1 = -1$ and $d_1 = 1$. At the k-th stage, the processors do the following: processor P_i transmits a message $m_{i,k} = 0$ if $(-1)^{i-1} f_i'((a_k + b_k)/2) \leq (c_k + d_k)/2$; otherwise it transmits $m_{i,k} = 1$.

If $m_{1,k} = 0$ and $m_{2,k} = 1$, then $f_1'((a_k + b_k)/2) + f_2'((a_k + b_k)/2) \leq 0$. We may then let $a_{k+1} = (a_k + b_k)/2$ and leave b_k, c_k, d_k unchanged. Similarly, if $m_{1,k} = 1$ and $m_{2,k} = 0$, we let $b_{k+1} = (a_k + b_k)/2$ and leave a_k, c_k, d_k unchanged.

We now consider the case $m_{1,k} = m_{2,k} = 1$. Let x^* be a minimizer of $f_1 + f_2$ belonging to $[a_k, b_k]$. If $x^* \geq (a_k + b_k)/2$, then $f_1'(x^*) \geq f_1'((a_k + b_k)/2) \geq (c_k + d_k)/2$. If $x^* \leq (a_k + b_k)/2$, then $f_1'(x^*) \geq -f_2'(x^*) \geq -f_2'((a_k + b_k)/2) \geq (c_k + d_k)/2$. In either case, we may let $c_{k+1} = (c_k + d_k)/2$ and leave a_k, b_k, d_k unchanged. Finally, if $m_{1,k} = m_{2,k} = 0$, a similar argument shows that we may let $d_{k+1} = (c_k + d_k)/2$ and leave a_k, b_k, c_k unchanged.

For each one of the four cases, we see that $a_k, ..., d_k$ will preserve properties (i), (ii) that were postulated earlier. Furthermore, at each stage, either $b_k - a_k$ or $d_k - c_k$ is halved. Therefore, after at most $k = 2\log(1/\epsilon)$ stages, we reach a point where either $b_k - a_k \leq \epsilon$ or $d_k - c_k \leq \epsilon$. If $b_k - a_k \leq \epsilon$, then there exists a minimizer which is within ϵ of a_k; given that the derivative of $f_1 + f_2$ is bounded by one, it follows that $f_1(a_k) + f_2(a_k)$ comes within ϵ of the optimum, as desired. Alternatively, if $d_k - c_k \leq \epsilon$, then $|f_1'((a_k + b_k)/2) + f_2'((a_k + b_k)/2)| \leq d_k - c_k \leq \epsilon$. It follows that for any $x \in [0, 1]$, we have $f_1(x) + f_2(x) \geq f_1((a_k + b_k)/2) + f_2((a_k + b_k)/2) - |x - (a_k + b_k)/2|\epsilon$, which shows that $(f_1 + f_2)((a_k + b_k)/2)$ comes within ϵ of the optimum. **Q.E.D.**

4.2.4 An Optimal Protocol for Strongly Convex Problems

We consider here the class $\Theta_{SC,M,L}$ of strongly convex problems, defined in Subsection 4.2.1. However, we prefer to deal with unconstrained optimization and for this reason, we limit ourseleves to to the subset Θ_U of $\Theta_{SC,M,L}$ which contains only functions for which there exists a (necessarily unique) $x^* \in [0, 1]^n$ at which $f'(x^*) = 0$. This assumption is not unrealistic; in practical unconstrained optimization problems one is able to obtain a priori bounds on the region in which the optimum is

lying. Then, by rescaling, this region may be assumed to be $[0,1]^n$. We show below that an appropriate inexact implementation of the classical gradient algorithm turns out to lead to an optimal protocol, as far as the dependence on n and ϵ is concerned.

The protocol produces a sequence $\{x_k\}_{k=0}^T$ of points according to the iteration

$$x_{k+1} = x_k - \gamma s_k; \qquad x_k = 0, \tag{4.2.6}$$

where s_k is an approximation to g_k, the gradient of $f_1 + f_2$, evaluated at x_k. In particular, we will require that the inequality

$$||s_k - g_k|| \leq n^{1/2}\alpha^k \tag{4.2.7}$$

holds for all $k \leq T$, with some $\alpha \in (0,1)$. (Throughout this section $||\cdot||$ will stand for the Euclidean norm.) We first estimate the number of iterations required to reach an ϵ–optimal point.

Proposition 4.2.5 *If the stepsize γ is small enough, and if α is sufficiently close to 1, then there exist $A > 0$, $B > 0$, depending only on M, L, such that*

(a) $f(x_k) - f^* \leq An\alpha^k$, (4.2.8)
(b) $||x_{k+1} - x_k|| \leq Bn^{1/2}\alpha^k$. (4.2.9)

Proof: We state without proof the following properties of functions in $\Theta_{SC,M,L}$ [NY 83, pages 254–255]:

(i) $||f'(x) - f'(y)|| \leq ML||x - y||$. (4.2.10)
(ii) $f(x+y) \geq f(x) + \langle f'(x)|y\rangle + (L/2)||y||^2$. (4.2.11)
(iii) $f(x+y) \leq f(x) + \langle f'(x)|y\rangle + (LM/2)||y||^2$. (4.2.12)

Using inequality (4.2.12) together with (4.2.6) and (4.2.7) we obtain

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) - \gamma\langle s_k, g_k\rangle + \gamma^2\frac{LM}{2}||s_k||^2 \\ &\leq f(x_k) - \gamma||g_k||^2 + \gamma n^{1/2}\alpha^k + \gamma^2\frac{LM}{2}(||g_k||^2 + n\alpha^{2k}). \end{aligned} \tag{4.2.13}$$

Let x^* be the minimizer of f. Using 4.2.1 we obtain $||g_k|| \geq L||x_k - x^*||$ and using (4.2.11) we obtain

$$f(x_k) - f(x^*) \leq \frac{ML}{2}||x_k - x^*||^2 \leq (M/2L)||g_k||^2.$$

Combining this with 4.2.13 we conclude that

$$\begin{aligned} f(x_{k+1}) - f(x^*) \;\leq\; & f(x_k) - f(x^*) - (\gamma 2L/M)(1 - \gamma LM/2)(f(x_k) - f(x^*)) \\ & + \gamma n^{1/2}\alpha^k + n\gamma^2(LM/2)\alpha^{2k}. \end{aligned}$$

Taking γ small enough so that $\beta = 1 - 2(\gamma L/M)(1 - \gamma LM/2) \in (0,1)$, and assuming that α was chosen larger than β, part (a) is easily proved by induction. (To start the induction, we use the inequality (4.2.12).) For part (b) we use inequality (4.2.11) to obtain

$$||x_k - x^*||^2 \leq \frac{2}{L}(f(x_k) - f(x^*))$$

and the result follows. **Q.E.D.**

As an immediate corollary of Proposition 4.2.5, we see that after $A(\log(1/\epsilon) + \log n)$ iterations of the approximate gradient algorithm (4.2.6)–(4.2.7), we have reached an ϵ–optimal point. Next we show how this algorithm may be implemented in a distributed manner with only $O(n)$ bits being communicated at each stage. All we need to do is to make sure that the processor share at each stage enough information to compute a vector s_k satisfying the bound (4.2.7). This is accomplished by having each processor know a set of scalars $s_k(i,j)$, $i = 1,2$, $j = 1,...,n$ such that $|s_k(i,j) - g_k(i,j)| \leq n^{1/2}\alpha^k$, where $g_k(i,j)$ is the j-th component of $f_i'(x_k)$. At stage $k = 0$ this is easy: $g_0(i,j)$ is bounded by $n^{1/2}$ and therefore the choice $s_0(i,j) = 0$ will do. Suppose that quantities $s_k(i,j)$ with the desired properties have been shared at stage k and let us now consider stage $k+1$. We have $|g_{k+1}(i,j) - s_k(i,j)| \leq |g_{k+1}(i,j) - g_k(i,j)| + |g_k(i,j) - s_k(i,j)| \leq LM||x_{k+1} - x_k|| + n^{1/2}\alpha^k \leq (LMB+1)n^{1/2}\alpha^k$, where the last inequality follows from part (b) of Proposition 4.2.5. We require that $s_{k+1}(i,j)$ be an integer multiple of $n^{1/2}\alpha^{k+1}$. This requirement does not prohibit the satisfaction of the constraint (4.2.7); furthermore, with this requirement, there are at most $(LMB+1)\alpha^{-1} + 1$ possible choices for $s_{k+1}(i,j)$. Therefore, processor P_i may choose $s_{k+1}(i,j)$ as above and transmit its value to the other processor while communicating only a constant number of bits. This has to be done by each processor P_i and for each component j, for a total of $O(n)$ bits of communication per stage. We have thus proved the following result.

Proposition 4.2.6 $C(\Theta_U;\epsilon) \leq An(\log n + \log(1/\epsilon))$, *where* A *is a constant depending only on* M, L.

Notice that this result attains the lower bound of Section 4.2.1.

4.3 One–Way vs. Two–Way Protocols

In this section, we compare the power of one–way communication protocols with that of two–way communication protocols. We show that two–way communication protocols can provide exponential savings over the one–way protocols (see [PS 82] for another proof of this fact). In particular, we consider a situation where two processors P_1 and P_2 are to determine if there exists a point $x^* \in [0,1]^n$ such that $\text{dist}(x^*, A_1) \leq \epsilon$ and $\text{dist}(x^*, A_2) \leq \epsilon$, where $0 < \epsilon < 1$ and A_1, A_2 are two convex sets in $[0,1]^n$, under the assumption that processor P_i has access to A_i only $(i = 1,2)$. We study the minimum number of binary bits that need to be exchanged between processors P_1 and P_2, as a function of ϵ. By applying the result of Section 4.2, we obtain a two–way communication protocol for this problem which uses at most $O(\log^2 \frac{1}{\epsilon})$ bits (see Subsection 4.3.1). We then use a result of Dudley, which gives bounds on the number of bits needed to approximate a convex set, and show that the one–way communication complexity for the same problem is $O(\epsilon^{-\frac{n}{2}} \log \frac{1}{\epsilon})$ (see Subsection 4.3.2). We also show a lower bound of $\Omega(\epsilon^{-\frac{n}{2}})$ for this problem. In Subsection 4.3.2, we also give a geometric proof of Dudley's result for the case $n = 2$.

4.3.1 A Two–Way Communication Protocol

Let there be two processors P_1 and P_2 who wish to decide if two convex sets A_1, A_2 in $[0,1]^n$ are disjoint, assuming that processor P_i has access to the convex set A_i only $(i = 1,2)$. The two processors proceed by exchanging messages, which are assumed to be binary strings of finite length, until one of them can decide if $A_1 \bigcap A_2$ is empty. Notice that there exist disjoint convex sets which are arbitrarily close and to decide if these convex sets are disjoint the two processors may have to exchange

an unbounded number of messages. In fact, one can simply consider the special case where A_1 and A_2 are two points in $[0,1]^n$. Clearly, the problem of deciding if two points (which may be arbitrarily close) of $[0,1]^n$ are the same, using discrete communication protocols, has unbounded two–way communication complexity. Thus, in order to ensure a finite communication complexity, an "approximate" notion of *disjointness* is needed. This is accomplished as follows.

Let $\epsilon \in (0,1)$ and let $\mathcal{G}_1$ be the set of all pairs (A_1, A_2) such that

(a) A_1, A_2 are convex subsets of $[0,1]^n$;
(b) $\text{dist}(A_1, A_2) \leq \epsilon$.

Similarly, we let $\mathcal{G}_2$ be the set of all pairs (A_1, A_2) satisfying (a) and

(b') $\text{dist}(A_1, A_2) \geq 2\epsilon$.

Clearly, $\mathcal{G}_1 \bigcap \mathcal{G}_2$ is empty. Roughly speaking, $\mathcal{G}_1$ (respectively, $\mathcal{G}_2$) contains all pairs of convex sets which are "almost" intersecting (respectively, disjoint). Let $\mathcal{G} = \mathcal{G}_1 \bigcup \mathcal{G}_2$. Consider now the following feasibility problem (P):

For each $(A_1, A_2) \in \mathcal{G}$, processors P_1, P_2 are to determine if $(A_1, A_2) \in \mathcal{G}_1$ or $(A_1, A_2) \in \mathcal{G}_2$, under the assumption that processor P_i knows the convex set A_i only $(i = 1, 2)$. If they decide that $(A_1, A_2) \in \mathcal{G}_1$, then they will provide a point x^* in $[0,1]^n$ (as a certificate) such that

$$\text{dist}(x^*, A_1) + \text{dist}(x^*, A_2) < 2\epsilon$$

where $\text{dist}(\cdot, \cdot)$ means the distance between the two sets.

It turns out that the above problem can be thought of as a special case of the convex optimization problem considered in Section 4.2. More specifically, if we define $f_1(x)$ to be $\text{dist}(x, A_1)$ and $f_2(x)$ to be $\text{dist}(x, A_2)$, then clearly f_1, f_2 are convex and also Lipschitz continuous. We claim that $(A_1, A_2) \in \mathcal{G}_2$ if and only if each ϵ–optimal value of the optimization problem

$$\min_{x \in [0,1]^n} (f_1(x) + f_2(x))$$

is less than 2ϵ. (Here, a real number c is called an ϵ–optimal value if $c \leq f_1(x) + f_2(x) + \epsilon$, for all $x \in [0,1]^n$. Similarly, a point $x^* \in [0,1]^n$ is called an ϵ–optimal

solution if $f(x^*)$ is an ϵ–optimal value.) In fact, since

$$\text{dist}(A_1, A_2) = \min_{x \in [0,1]^n} \{\text{dist}(x, A_1) + \text{dist}(x, A_2)\},$$

it follows that any ϵ-optimal solution x^* to the above given optimization problem will satisfy

$$\text{dist}(x^*, A_1) + \text{dist}(x^*, A_2) \leq \text{dist}(A_1, A_2) + \epsilon.$$

Therefore, using the fact that $\text{dist}(A_1, A_2)$ is either less than ϵ or greater than 2ϵ, we see that $\text{dist}(x, A_1) + \text{dist}(x, A_2) < 2\epsilon$ iff $\text{dist}(A_1, A_2) \leq 2\epsilon$. Thus, problem (P) has been reduced to the problem of finding an ϵ–optimal solution to the optimization problem given above. Consequently, problem (P) can be solved by the general protocol described in Subsection 4.2.2, with communication complexity $O(\log^2 \frac{1}{\epsilon})$.

4.3.2 One–Way Communication Complexity

By symmetry, we only consider the one way communication protocols where messages are sent from processor P_1 to processor P_2. Since processor P_2 has no access to the convex set A_1 it is both necessary and sufficient for P_1 to send an ϵ–approximate version of A_1 to processor P_2, in order for P_2 to be able to determine if $A_1 \bigcap A_2$ is empty. More precisely, we define the variation between A_1, A to be the following,

$$\rho(A_1, A) = \max\{\sup_{x \in \partial(A_1)} \inf_{y \in \partial(A)} \text{dist}(x, y), \sup_{y \in \partial(A)} \inf_{x \in \partial(A_1)} \text{dist}(x, y)\}, \qquad (4.3.1)$$

where $\partial(A_1)$ and $\partial(A)$ denote the boundaries of A and A_1. The left hand side of 4.3.1 is equal to the so called Hausdorff distance between $\partial(A_1)$ and $\partial(A)$. ([M 75] contains the definition and some properties of the Hausdorff metric.) Clearly, ρ is symmetric in the sense that $\rho(A_1, A) = \rho(A, A_1)$. We will say that A_1, A are more than ϵ apart if $\rho(A_1, A) \geq \epsilon$.

Note that processor P_1 does not know the location of the convex set A_2. It follows that P_1 should send to processor P_2 a convex set A such that $\rho(A_1, A) \leq \epsilon$, because otherwise P_2 would not have enough information to determine if $\text{dist}(A_1, A_2) < \epsilon$. On the other hand, sending an ϵ–approximation of A_1 is also sufficient. In fact, by the choice of A, we see that

$$\text{dist}(A_1, A_2) - \epsilon \leq \text{dist}(A, A_2) \leq \text{dist}(A_1, A_2) + \epsilon.$$

This, together with the fact that (A_1, A_2) is either in $\mathcal{G}_1$ or in $\mathcal{G}_2$, implies that $\text{dist}(A_1, A_2) < \epsilon$ iff $\text{dist}(A, A_2) < 2\epsilon$. Thus, problem (P) has been reduced to the problem of how to represent a convex set in $[0,1]^n$ within ϵ accuracy using as few binary bits as possible. The following result is due to R.M. Dudley ([D 74]) who provided tight upper and lower bounds on the number of bits required to represent a convex set in $[0,1]^n$ within ϵ accuracy.

Let $\Lambda(n)$ denote the class of convex sets in $[0,1]^n$. Let $N_n(\epsilon)$ be the number of convex sets needed to form an ϵ–dense set in $\Lambda(n)$, i.e., to approximate each set within ϵ for the Hausdorff metric.

Theorem 4.3.1 [Dudley] *Given any* $r < \frac{n-1}{2} < s$, *then* $\exp(\epsilon^{-r}) < N_n(\epsilon) < \exp(\epsilon^{-s})$, *for* ϵ *small enough.*

As an immediate corollary of Theorem 4.3.1 and the results in Subsection 4.3.1, we conclude:

Corollary 4.3.1 *The one–way communication complexity for solving* (P) *is at least* $\Omega(\epsilon^{-t})$, *for any* $t > \frac{n-1}{2}$, *while the two–way communication complexity is at most* $O(\log^2 \frac{1}{\epsilon})$. *Thus, two–way communication protocols can provide exponential savings for solving* (P).

In what follows, we give a different and simpler proof of Dudley's result for the case $n = 2$. Our approach is geometric in nature and can be extended to work for general n. In addition, we also give a new proof of Dudley's lower bound. But before we start, let us first give a naive upper bound on the number of bits needed to represent a convex set in $[0,1]^n$ with ϵ accuracy.

A Naive Upper Bound

We divide the unit cube $[0,1]^n$ into $(1/c\epsilon)^n$ smaller cubes by dividing each dimension into $1/c\epsilon$ equal subintervals, where c is a positive constant to be determined later. Given a convex body A_1 in $[0,1]^n$, consider the set of cubes that intersect A_1. Let us call the convex hull of these cubes A. In what follows we show that $\rho(A_1, A) \leq \epsilon$.

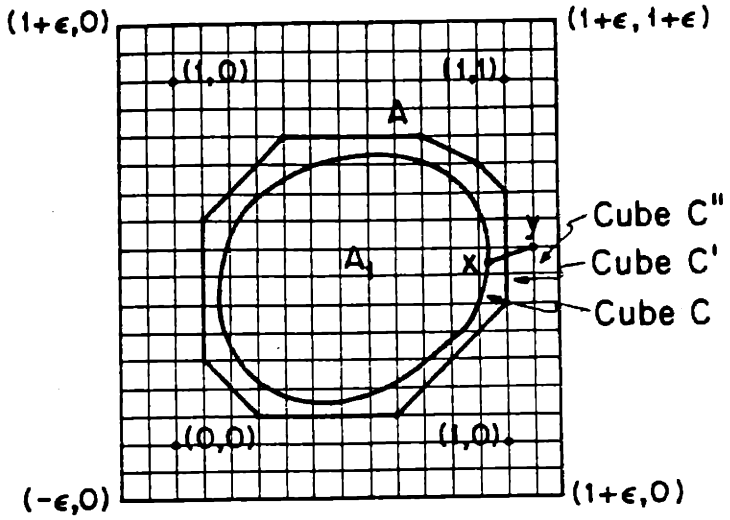

Figure 4.1: A naive way to approximate a convex set.

Obviously, we have $A_1 \subset A$. Now for any point $x \in \partial(A_1)$, let C be the cube in which the point x falls (see Figure 4.1). Consider all the neighboring cubes of C and notice that at least one of the these cubes is not contained in A_1. (If A_1 contains all the neighboring cubes of C, then C would have been completely inside A_1, by the convexity of the set A_1. Moreover, x would have to be an interior point of A_1 which contradicts the assumption $x \in \partial(A_1)$). Now let C' be a neighboring cube of C which is not contained in A_1. Then, one of the neighboring cubes of C' will not intersect A_1, for otherwise C' would be inside A_1. Let C'' be a cube which does not intersect A_1. Then by the way A is constructed we know that C'' does not intersect A either. We take any corner of C'', say y, and consider the line segment joining x and y. Clearly the segment will intersect ∂A and therefore

$$\text{dist}(x, y) \geq \text{dist}(x, \partial(A)).$$

However, it is easily seen that $\text{dist}(x, y) \leq \epsilon$, if c is chosen appropriately, therefore $\text{dist}(x, \partial(A_1)) \leq \epsilon$ for all $x \in \partial(A_1)$. Hence

$$\rho(A, A_1) = \sup_{x \in \partial(A_1)} \text{dist}(x, \partial(A)) \leq \epsilon.$$

Next we show that one can represent A with $O((\frac{1}{\epsilon})^{n-1}\log\frac{1}{\epsilon})$ bits. Notice that it is sufficient to just code the vertices of A. Note that each vertex of A is a corner of some cube in $[0,1]^n$, thus it can be represented with $O(\log\frac{1}{\epsilon})$ bits. (The dimension n is viewed as a constant in this subsection.) Now the problem becomes to count the number of vertices that A can have.

Suppose x_0 is an interior point of A_1. Let us consider the set of points

$$A' = \left\{ x |\ x = x_0 + \lambda(y - x_0),\ y \in \partial A_1,\ 0 \leq \lambda \leq 1 + \frac{c_1\epsilon}{||y - x_0||} \right\}$$

It is clear that when c_1 is appropriately chosen (independent of ϵ) we will have $A \subset A'$ and the volume of the set $A' \backslash A_1$ can be bounded by $\text{Area}(\partial A') \times c_1\epsilon$.

Since $A' \subset [0, 1 + c_1\epsilon]^n$ and A' is convex, it follows that

$$\text{Area}(\partial(A_1)) \leq \text{Area}(\partial([0, 1 + c_1\epsilon]^n)) = O(1)$$

Therefore the volume of the set $A' \backslash A$ is less than $O(\epsilon)$. Since each cube has volume $(c\epsilon)^n$ and they do not overlap we conclude that there can be at most $O((\frac{1}{\epsilon})^{n-1})$ cubes inside the set $A' \backslash A_1$. Since each cube inside $A' \backslash A_1$ can have at most $2n$ neighboring cubes we know that the total number of grid points inside $A' \backslash A_1$ can be at most $2n2^n \times O((\frac{1}{\epsilon})^{n-1}) = O((\frac{1}{\epsilon})^{n-1})$. Since the vertices of A all lie in the set $A' \backslash A_1$, it follows that the total number of vertices of A can be at most $O((\frac{1}{\epsilon})^{n-1})$. Therefore we have just shown the following theorem.

Proposition 4.3.1 *There exists a one way communication protocol for solving the problem (P) which uses at most $O((\frac{1}{\epsilon})^{n-1}\log\frac{1}{\epsilon})$ bits.*

By comparing to Theorem 4.3.1, we see that Propositon 4.3.1 is far from being tight.

A New Proof of Dudley's Lower Bound

Next, we show a lower bound on the number of binary bits needed to represent a convex set in $[0,1]^n$ within ϵ accuracy and thus establish lower bounds on the one

way communication complexity for solving the problem (P) described in Subsection 4.3.1. In the case $n = 1$, the trivial lower bound $\Omega(\log \frac{1}{\epsilon})$ matches the obvious upper bound. For $n \geq 2$, we show a lower bound of $\Omega((\frac{1}{\epsilon})^{\frac{n-1}{2}})$. Our approach is to construct a family $S(\epsilon)$ of convex sets $\{A_i\}$ in $[0,1]^n$ such that any two of the convex sets A_i, A_j $(i \neq j)$ are ϵ apart, in other words, $\rho(A_i, A_j) \geq \epsilon$. Clearly, any protocol that solves the problem (P) will have to assign different information sequences to these convex sets in order to be able to differentiate these convex sets. Therefore we have a lower bound $\log |S(\epsilon)|$ on the number of bits required to solve (P).

Let us assume $n \geq 2$. For convenience we will consider the cube $[-1,1]^n$ instead of $[0,1]^n$. This modification is inessential since we easily translate the result for the cube $[-1,1]^n$ to a result for $[0,1]^n$, by using a scaling argument. First we note that the unit ball is inside the cube $[-1,1]^n$. Given any $\delta \in (0,1)$, consider the following set of points on the surface of the unit ball,

$$x = \left(k_1 c_2 \delta,\ k_2 c_2 \delta,\ \ldots,\ k_{n-1} c_2 \delta,\ c_2 \delta \sqrt{\frac{1}{(c_2\delta)^2} - k_1^2 - \cdots - k_{n-1}^2}\right) \tag{4.3.2}$$

Here c_2 is a constant to be chosen later and $k_1, k_2, \ldots, k_{n-1}$ are integers that satisfy the following relation

$$k_1^2 + \cdots + k_{n-1}^2 \leq \frac{1}{(c_2\delta)^2}$$

In other words, $(k_1, k_2, \ldots, k_{n-1})$ are the integer points inside the $n-1$ dimensional ball centered at the origin with radius $\frac{1}{c_2\delta}$. Since the volume of this ball is $\Omega((\frac{1}{\delta})^{n-1})$ (the constant only depends on n) we know that there must be $\Omega((\frac{1}{\delta})^{n-1})$ integer points inside the ball. In other words, there exist $\Omega((\frac{1}{\delta})^{n-1})$ points of the form (4.3.2) on the surface of the unit ball in $\Re^n$. The following lemma concerns the relation between the points defined by Eq. (4.3.2).

Lemma 4.3.1 *Let x_i, x_j, x_k be any three points defined by (4.3.2) such that $x_i \neq x_j$. Then,*

$$||x_i - x_j|| \geq c_2\delta \tag{4.3.3}$$

and

$$(x_i - x_k, x_j - x_k) \geq -(1 - c_3(c_2\delta)^2)||x_i - x_k||\ ||x_j - x_k|| \tag{4.3.4}$$

where c_3 is a constant that only depends on the dimension n.

Proof: Eq. (4.3.3) is obvious since if $x_i \neq x_j$ then they differ at some coordinate and therefore they are at least $c_2\delta$ apart in that coordinate.

Eq. (4.3.4) is a little trickier. First of all, by an easy calculation, we observe that (4.3.4) holds when dimension $n = 3$ for any x_i, x_j, x_k that are mutually more than $c_2\delta$ apart and satisfying $||x_i|| = ||x_j|| = ||x_k|| = 1$. If $x_k = x_i$ or $x_k = x_j$ then (4.3.4) is trivial. For $n \geq 3$, we know that there exists an linear orthonormal transformation T such that the coordinates of Tx_i, Tx_j, Tx_k are all zero except the first three coordinates. If $x_k = x_i$ or $x_k = x_j$ then (4.3.4) is again easy. If, however, $x_k \neq x_i$ and $x_k \neq x_j$ then by Eq. (4.3.3) x_i, x_j, x_k are mutually more than $c_2\delta$ apart. Therefore Tx_i, Tx_j, Tx_k are also more than $c_2\delta$ apart. Since the relation (4.3.4) holds for $n = 3$ we have

$$(Tx_i - Tx_k, Tx_j - Tx_k) \geq -(1 - c_3(c_2\delta)^2)||Tx_i - Tx_k||\ ||Tx_j - Tx_k||$$

Since orthonormal transformations preserve inner products and norms we see that Eq. (4.3.4) follows immediately from the above inequality. **Q.E.D.**

Let us now consider the convex hull of the points defined by equation (4.3.2). We first introduce some notations. We call the set of points defined by equation (4.3.2) $x_1, x_2, \ldots, x_m$. Let $CL(\Theta)$ denote the convex hull of a set of points Θ. For example, $CL(x_1, x_2, \ldots, x_m)$ will denote the convex hull of the points $x_1, x_2, \ldots, x_m$. Clearly, $CL(x_1, x_2, \ldots, x_m)$ is an approximation of the unit ball in R^n. The following lemma asserts that the distance of x_1 from $CL(x_2, \ldots, x_m)$ is at least δ^2, when the constant c_2 is appropriately chosen.

Lemma 4.3.2 *For sufficiently large c_2, there holds*

$$\text{dist}(x_1, CL(x_2, \ldots, x_m)) \geq \delta^2. \tag{4.3.5}$$

Proof: Given any point in $CL(x_2, \ldots, x_m)$, we can write it as $\sum_{i=2}^{l} \lambda_i x_i$, where $\sum_{i=2}^{l} \lambda_i = 1$ and $\lambda_i \geq 0$. By Caratheodory's theorem (see [SCH 86, page 94]) it suffices to assume $l \leq n + 1$. Consider the following relation

$$||x_1 - \sum_{i=2}^{l} \lambda_i x_i||^2 \ = \ ||\sum_{i=2}^{l} \lambda_i (x_i - x_1)||^2$$

$$
\begin{aligned}
&= \sum_{i=2}^{l} ||\lambda_i(x_i - x_1)||^2 + 2 \sum_{2\le i<j\le m} \lambda_i\lambda_j(x_i - x_1, x_j - x_1) \\
&\geq \sum_{i=2}^{l} ||\lambda_i(x_i - x_1)||^2 && (4.3.6) \\
&\quad -2 \sum_{2\le i<j\le l} \lambda_i\lambda_j||x_i - x_1||\ ||x_j - x_1||(1 - c_3(c_2\delta)^2) && (4.3.7) \\
&= (1 - c_3(c_2\delta)^2)(\sum_{i=2}^{l} ||\lambda_i(x_i - x_1)||^2 \\
&\quad -2 \sum_{2\le i<j\le l} \lambda_i\lambda_j||x_i - x_1||\ ||x_j - x_1||) + c_3(c_2\delta)^2 \sum_{i=2}^{l} ||\lambda_i(x_i - x_1)||^2 \\
&= 0.5(1 - c_3(c_2\delta)^2) \sum_{i,j=2}^{l} (\lambda_i||x_i - x_1|| - \lambda_j||x_j - x_1||)^2 \\
&\quad +c_3(c_2\delta)^2 \sum_{i=2}^{l} ||\lambda_i(x_i - x_1)||^2 && (4.3.8) \\
&\geq c_3(c_2\delta)^2 \sum_{i=2}^{l} ||\lambda_i(x_i - x_1)||^2 && (4.3.9) \\
&\geq c_3(c_2\delta)^2 \times \frac{\delta^2}{n+1} && (4.3.10) \\
&\geq \delta^4.
\end{aligned}
$$

Step (4.3.7) follows from equation (4.3.4). Step (4.3.8) is an algebraic identity. Step (4.3.9) is a consequence of equation (4.3.3) and the fact that $\sum_{i=2}^{l} \lambda_i = 1$ and $\lambda_i \geq 0$. The last step is true when the constant c_2 is chosen large enough. **Q.E.D.**

Corollary 4.3.2 *For any two distinct subsets Θ_1, Θ_2 of $\{x_1, \ldots, x_m\}$, we have*

$$\rho(CL(\Theta_1), CL(\Theta_2)) \geq \delta^2.$$

Proof: Without loss of generality we can assume that $x_1 \in \Theta_1$ and $x_1 \notin \Theta_2$. By Eq. (4.3.5), we see that

$$\text{dist}(x_1, CL(x_2, x_3, \ldots, x_m)) \geq \delta^2.$$

Since $\Theta_2 \subset \{x_2, x_3, \ldots, x_m\}$, we see that

$$\text{dist}(x_1, CL(\Theta_2)) \geq \delta^2,$$

from which the corollary immediately follows. **Q.E.D.**

If we take δ to be $\sqrt{\epsilon}$, then Corollary 4.3.2 implies that

$$\text{dist}(CL(\Theta_1), CL(\Theta_2)) \geq \epsilon,$$

for all distinct subsets Θ_1, Θ_2 of $\{x_1, x_2, \ldots, x_m\}$. It is well known that the number of different subsets of a set with cardinality m is 2^m. Since $m = \Omega((\frac{1}{\delta})^{n-1})$, we thus have shown the following theorem:

Theorem 4.3.2 *There holds* $|S(\epsilon)| \geq 2^{(\frac{1}{\epsilon})^{\frac{n-1}{2}}}$. *Thus, the size of any* ϵ*–dense set of* $\Lambda(n)$ *is at least* $\Omega(2^{(\frac{1}{\epsilon})^{\frac{n-1}{2}}})$.

Notice that for $n = 2$ the above result translates to an $\Omega(2^{\frac{1}{\sqrt{\epsilon}}})$ lower bound on the number of convex sets in $[0,1]^2$ which are mutually ϵ–apart and therefore to an $\Omega(\frac{1}{\sqrt{\epsilon}})$ lower bound on one–way communication complexity. The upper bound $O(\frac{1}{\epsilon})$ obtained from Proposition 4.2.2 is quite far from this lower bound. In what follows, we give an improved upper bound which comes very close to this lower bound. In particular, we prove that one can approximate an arbitrary convex set in $[0,1]^2$ with at most $O(\frac{1}{\sqrt{\epsilon}} \log \epsilon)$ bits, which is optimal within a logarithmic factor.

An Almost Tight Upper Bound for $n = 2$

In what follows, we present an improved method for representing convex sets in $[0,1]^2$ which requires only $O(\frac{1}{\sqrt{\epsilon}} \log \epsilon)$ binary bits in the worst case. By our early discussion, this method will lead to an $O(\frac{1}{\sqrt{\epsilon}} \log \epsilon)$ one–way communication protocol for solving problem (P) posed in Subsection 4.3.1.

Given a convex set A in $[0,1]^2$, we only need to approximate the boundary of A. Let us first introduce some notations. Suppose x_i for $i = 1, 2, \ldots, N$ are a sequence of adjacent points along the boundary of A (see Figure 4.2). We let l_i denote a line that supports A at the point x_i. For $i = 1, \ldots, N-1$, We will use β_i to denote the angle ($< \pi$) between l_i and l_{i+1} and d_i to denote the distance between x_i and x_{i+1}. We use β_N to denote the angle between l_N and l_1 and use d_N to denote the distance between x_N and x_1.

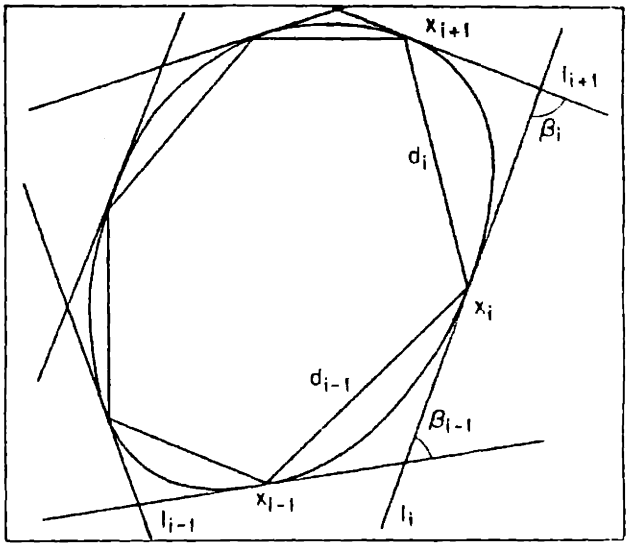

Figure 4.2: Approximating a convex set by a polygon.

Two simple observations are in order:

$$\beta_1 + \beta_2 + \cdots + \beta_N = 2\pi, \tag{4.3.11}$$

$$d_1 + d_2 + \cdots + d_N \leq 4. \tag{4.3.12}$$

Now suppose that we have found a sequence of x_i's $(i = 1, 2 \ldots, N+1)$ along $\partial(A)$ such that

$$d_i \beta_i \leq c_4 \epsilon, \tag{4.3.13}$$

where c_4 is constant to determined later.

Since the convex set A lies in one of the half–spaces defined by l_i we know that A lies in the intersection of these half spaces, which we call B. We claim that the convex set B is an $\frac{\epsilon}{2}$-approximation of A, when c_4 is chosen to be sufficiently small.

Lemma 4.3.3 *If $\beta_i \leq \pi/4$, $i = 1, \ldots, N$, and Eq. (4.3.13) holds, then we have*

$$\rho(A, B) \leq \frac{1}{2}\epsilon, \tag{4.3.14}$$

provided that c_4 is small enough.

Proof: Let y_i be the point where l_i and l_{i+1} intersects, $i = 1, 2, \ldots, N-1$ and l_N be the intersection point of l_N and l_1. Since A is convex it is obvious that B is the convex hull of y_i, $i = 1, 2, \ldots, N$. Let the distance from y_i to the line segment $\overline{x_i x_{i+1}}$ be t_i, $i = 1, 2, \ldots, N$. It is clear that to prove the relation (4.3.14) it suffices to prove $t_i \leq \frac{1}{2}\epsilon$ for all i. Now we notice that

$$\begin{aligned} t_i &\leq \tan \beta_i d_i \\ &\leq 2\beta_i d_i \qquad (4.3.15) \\ &\leq 2 \times \frac{1}{2} c_4 \epsilon \qquad (4.3.16) \\ &\leq \frac{1}{2}\epsilon. \end{aligned}$$

Step (4.3.15) is true when $\beta_i \leq \pi/4$. Step (4.3.16) is by assumption. The last step holds when the constant c_4 is small. **Q.E.D.**

The next lemma is essential for counting how many points x_i we need to use in approximating A to guarantee the relation (4.3.13) holds.

Lemma 4.3.4 *Given a sequence of points $x_1, \ldots, x_N$ on $\partial(A)$, if the relation*

$$d_i \beta_i \geq \frac{1}{2} c_4 \epsilon \qquad (4.3.17)$$

holds for more than $N/2$ different i's, then we have

$$N \leq (2 + \pi) \sqrt{\frac{1}{2c_4}} \frac{1}{\sqrt{\epsilon}}.$$

Proof: Let I denote the set of indices for whcih the relation (4.3.17) holds. It is clear that for $i \in I$ we have

$$\begin{aligned} \sqrt{\frac{1}{2} c_4 \epsilon} &\leq \sqrt{\beta_i d_i} \\ &\leq \frac{\beta_i + d_i}{2}. \end{aligned}$$

Summing above inequalities over all $i \in I$ and using Eqs. (4.3.11)–(4.3.12), we obtain

$$\frac{N}{2} \sqrt{\frac{1}{2} c_4 \epsilon} \leq \frac{1}{2} \sum_{i \in I} \beta_i + \frac{1}{2} \sum_{i \in I} d_i$$

$$\begin{aligned} &\leq \frac{1}{2}\sum_i \beta_i + \frac{1}{2}\sum_i d_i \\ &\leq \frac{4}{2} + \frac{2\pi}{2} \\ &= 2 + \pi. \end{aligned}$$

Therefore we have

$$N \leq (2+\pi)\sqrt{\frac{1}{2c_4}}\frac{1}{\sqrt{\epsilon}},$$

which proves the lemma. **Q.E.D.**

In summary, we have shown that if we can find a sequence of points $x_1, \ldots, x_N$ on $\partial(A)$ satisfying (4.3.13) and $\beta_i \leq \pi/4$ for all i, then we can construct a convex set B that is an $\frac{\epsilon}{2}$–approximaton of A. If, in addition, the x_i's also satisfy the relation (4.3.17) for more than $N/2$ different i's, then B will have at most $O(\frac{1}{\sqrt{\epsilon}})$ corners. To code such a convex set, we only need to code its corners and slopes of the boundary lines. Since there are no more than $O(\frac{1}{\sqrt{\epsilon}})$ corners and boundary lines and each of them can be represented within $O(\epsilon)$ accuracy with at most $O(\log \frac{1}{\epsilon})$ bits, we can code the convex set B within $\frac{\epsilon}{2}$ accuracy with $O(\frac{1}{\sqrt{\epsilon}} \log \frac{1}{\epsilon})$ bits. Therefore, we can represent the convex set A in ϵ accuracy with $O(\frac{1}{\sqrt{\epsilon}} \log \frac{1}{\epsilon})$ binary digits, provided that we can find a sequence $x_1, \ldots, x_N$ on $\partial(A)$ satisfying the constraint $\beta_i \leq \pi/4$ for all i, (4.3.13), and (4.3.17) for at least $N/2$ different i's.

We now turn to the issue of constructing sequences of points $x_1, \ldots, x_n$ that satisfy these conditions. In fact, there are many ways to construct such sequences. Below is a construction that is particularly simple. We first find eight points $x_1, \ldots x_8$ on $\partial(A)$ and eight lines $l_1, \ldots, l_8$ such that: (a) l_i is a supporting line of A at the point x_i; (b) each angle between the lines l_i, l_{i+1} $(1 \leq i \leq 7)$ is equal to $\pi/4$. As a result, the angle between l_8 and l_1 is also equal to $\pi/4$. (This is possible since we do not require the points to be distinct.) Then $\partial(A)$ is partitioned (by the eight points) into eight segments. At a general step of the construction, the boundary $\partial(A)$ is partitioned into a number of segments. Now we can pick any segment and let z_1, z_2 be its endpoints and l_1, l_2 be the tangents at z_1, z_2 respectively. Let the angle between l_1, l_2 be β and the distance between z_1, z_2 be d (see Figure 3). If $\beta d > c_4\epsilon$, then we find a point z_3 on $\partial(A)$ such that

$$\beta_{13} d_{13} = \beta_{23} d_{23} \tag{4.3.18}$$

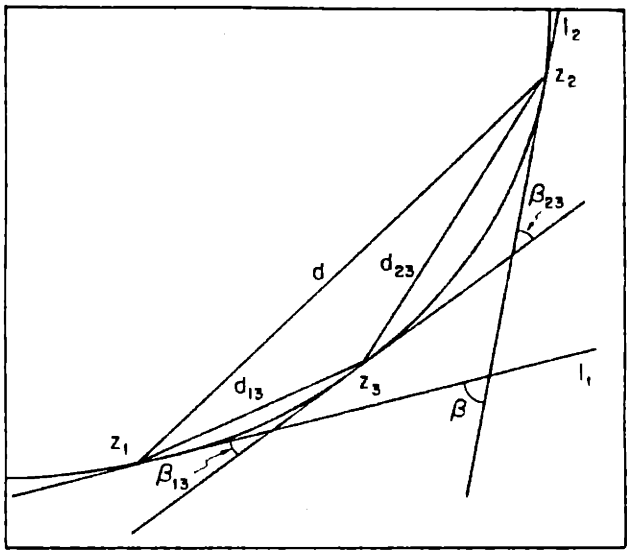

Figure 4.3: Construction of a sequence of points $x_1, \ldots, x_N$ on $\partial(A)$.

where β_{13} denotes the angle between l_1 and l_3 (the tangent line at z_3) and β_{23} denotes the angle between l_2 and l_3, d_{13}, d_{23} denote the distances between z_1, z_3 and z_2, z_3 respectively. As z_3 approaches z_1, we know that $\beta_{13}d_{13}$ goes to zero and $\beta_{23}d_{23}$ goes to βd. Similarly as z_3 approaches z_2, $\beta_{13}d_{13}$ converges to βd and $\beta_{23}d_{23}$ converges to zero. Therefore, by continuity, there exists a point z_3 in the segment satisfying the relation (4.3.18). Now there are two cases:

Case 1:

$$\beta_{13}d_{13} = \beta_{23}d_{23} \geq \frac{c_4}{2}\epsilon$$

In this case, we simply take z_3 as a new point in our final sequence. $\partial(A)$ is then partitioned into one more segment by the presence of z_3.

Case 2:

$$\beta_{13}d_{13} = \beta_{23}d_{23} < \frac{c_4}{2}\epsilon$$

Then by a similar continuity argument there exists a point $z_3' \in \partial(A)$ such that both $\beta_{13}d_{13}$ and $\beta_{23}d_{23}$ are less than $c_4\epsilon$ and at least one of $\beta_{13}d_{13}, \beta_{23}d_{23}$ is greater than $\frac{c_4}{2}\epsilon$. Now the segment (with z_1, z_2 being its endpoints) is partitioned into two segments that satisfy (4.3.13) and at least one of which satisfies the relation (4.3.17).

We repeat this process as long as there exists a segment for which $\beta d > c_4\epsilon$. Notice that this process has to terminate at some point due to Eqs. (4.3.11) and (4.3.12). In the end, we will have found a sequence of points that satisfy the relation (4.3.13) and at least half of the segments satisfy the relation (4.3.17).

By applying Lemmas 4.3.3 and 4.3.4 to the sequence of points $\{x_i\}$ we have just constructed, we obtain the following:

Theorem 4.3.3 *We can represent each convex set A in $[0,1]^2$ with at most $O(\frac{1}{\sqrt{\epsilon}}\log\frac{1}{\epsilon})$ bits. Consequently, there exists an one-way communication protocol for solving problem (P) that requires at most $O(\frac{1}{\sqrt{\epsilon}}\log\frac{1}{\epsilon})$ bits of information exchange in the worst case.*

4.4 Discussion and Possible Extensions

In this chapter, we have studied two-way discrete communication complexity. First of all, we have pointed out that Abelson's result, which gives lower bound on the two-way continuous communication complexity, remains valid for problem of computing a Boolean function f with a discrete communication protocol, provided that we view f as a polynomial on an appropriately defined field $\mathcal{F}$. In fact, we have shown that each discrete two-way communication protocol can be viewed as a purely polynomial protocol over the field $\mathcal{F}$. Then, we turned to the problem of approximately minimizing the sum of two convex functions f_1, f_2, defined on $[0,1]^n$. In particular, we considered a situation where two processors P_1, P_2 wish to find an ϵ-optimal solution x^* to the problem $\min_{x\in[0,1]^n}(f_1(x)+f_2(x))$ (i.e., $f_1(x^*)+f_2(x^*) \leq f_1(x)+f_2(x)+\epsilon$, for all $x \in [0,1]^n$), assuming that processor P_i has access to the function f_i only $(i=1)$. We have obtained several upper bounds for the minimum number bits that need to be exchanged between the two processors in order to find an ϵ-optimal solution. We have also derived lower bounds which come quite close to these upper bounds. As an application of these result, we have considered the problem of deciding if two convex subsets in $[0,1]^n$ are (almost) disjoint, and proved that two-way communication protocols can be exponentially more powerful than one-way communication protocols, which reestablished a re-

sult by Papadimitriou and Sipser [PS 82]. In addition, we have shown how discrete one–way communication protocols for this problem are related to the problem of representing a convex set with binary digits, which was studied by Dudley [D 74]. We have also given a simple geometric proof of Dudley's result for the case $n = 2$.

There are many important unsettled questions in this area. For instance, at the end of Section 4.1, we have given an example for which Theorem 4.1.1 can provide a tight lower bound. It is of interest to test Theorem 4.1.1 on some other nontrivial Boolean functions. More importantly, it remains to be seen how Theorem 4.1.1 compares with other lower bounds by Yao and other researchers ([Y 79], [MS 82] and etc.). For the problem of distributed convex optimization, a couple of open problems are in order:

1. The protocol of Subsection 4.2.4 is likely to be far from optimal concerning the dependence on the parameters M and L. The gradient algorithm tends to be inefficient for poorly conditioned problems (large M), whereas a version of the conjugate gradient method requires an optimal number of iterations [NY 83]. It remains to be seen whether a suitable approximate version of the conjugate gradient method admits a distributed implementation with low communication requirements.

2. In Subsection 4.2.4 we dealt essentially with unconstrained problems for which the optimal solution may be a priori localized into a bounded region. If we consider true constrained problems, we expect that a similar approximate version of the gradient projection method will have the same communication requirements as the gradient method.

3. For the class Θ_L, gradient methods do not work and the gap between the lower bound of Subsection 4.2.1 and the upper bound of Subsection 4.2.2 remains open. We believe that the factor of n^2 in the upper bound cannot be reduced. The reason is that any conceivable algorithm would need to consider at least $O(n \log(1/\epsilon))$ points and it is hard to imagine of any useful transfer of information concerning the behavior of the function in the vicinity of a point which does not require $O(n)$ bits. On the other hand, it may be possible to reduce the factor $\log^2(1/\epsilon)$ to just $\log(1/\epsilon)$ although we do not know how to accomplish this.

4. Another extension concerns the case of $K > 2$ processors minimizing the function

$f_1 + \cdots + f_K$. The protocols of Propositions 4.2.4 and 4.2.6 may be adjusted to obtain protocols with communication $O(K \log K)$, as far as the dependence on K is concerned. It is an open question whether this may be reduced to just $O(K)$.

Before we close this chapter, we make a remark on Dudley's result (cf. Theorem 4.3.1). In his original proof, Dudley actually showed a stronger result, namely, that for each convex set A in $[0,1]^n$, there exists a polygon B which is the intersection of at most $O(\epsilon^{\frac{n-1}{2}})$ half–spaces, such that $\mathrm{dist}(A,B) \leq \epsilon$, where $\mathrm{dist}(\cdot,\cdot)$ denotes the Hausdorff metric. (Note that the results in Section 4.3.2 also established this fact for the case $n = 2$.) Thus, by combining this stronger result with Corollary 4.3.2, we see that the minimum number of hyperplanes needed to approximate a convex subset A of $[0,1]^n$ within ϵ accuracy is $\Theta(\epsilon^{\frac{n-1}{2}})$.

Chapter 5

Multi–Party Communication Protocols

As we mentioned in Chapter 1, very little is known about multi–party communication protocols and so far most of the research has been devoted to the study of the two–party case. In this chapter, we consider a situation where a collection of processors $P_1, \ldots, P_n$ are to compute some function $f(x_1, x_2, \ldots, x_n)$, assuming that each processor P_i has access only to the variable x_i and the functional form of f. The processors are linked through a network of communication channels described by a connected and undirected graph $G = (V, E)$. The nodes of G correspond to the processors $P_1, \ldots, P_n$, and the edges of G are used to describe the communication channels of the network. We consider two kinds of communication protocols, namely, continuous protocols and discrete protocols. In Section 5.1, continuous protocols, in which messages are real–valued functions are considered. In particular, we demonstrate that the problem of multi–party communication using such protocols is in general combinatorially intractable. More specifically, we show that the problem of designing an optimal protocol to compute even a linear function is NP–Complete. In Section 5.2, we consider a restricted class of multi–party communication protocols in which messages are binary strings and the communication channels are assumed to have a tree structure. For such restricted protocols, we prove an almost tight lower bound (within a constant factor) on the number of bits that must be transmitted in order for the processors $P_1, \ldots, P_n$ to compute a linear function $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i$ within a certain accuracy. A similar bound is also established for the function $f(x_1, \ldots, x_n) = \prod_{i=1}^{n} x_i$. In the final section (Section 5.3), we discuss the results obtained in this chapter and describe some open problems for future study.

5.1 A Negative Result

In this section, we show that the problem of designing an optimal multi–party communication protocol for computing a linear function is computationally intractable. As seen in the previous chapters, this problem is trivial in the case of two–party communication. In fact, if there are only two parties involved in the communication, we only need to take into account the analytic properties of $f(x, y)$ (the function to be computed) when we are designing an optimal protocol. In the case of multi–party communication, new challenges arise as a result of combinatorial structure of the network $G = (V, E)$. In fact, even if the content of each message is determined, the questions related to message routing still remain. (Notice that such issues are nonexistent in the case of two–party communication.) In what follows, we isolate these combinatorial issues by letting f have the simplest possible analytic form: $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} a_i x_i$ ($x_i \in \Re$, $a_i \in \Re$, for all i), and we show that the design of an optimal protocol is still intractable. Thus, our result may, to some extent, explain the lack of research progress in the study of multi–party communication complexity.

5.1.1 Multi–Party Protocols, Continuous Case

Let $f : D_{x_1} \times \cdots \times D_{x_n} \mapsto \Re$ be some function, where each D_{x_i} is some open subset of $\Re^{d_i}$. We consider a situation where a collection of n processors $P_1, \ldots, P_n$ are to compute f, assuming that processor P_i only knows the functional form of f and the value of the variable $x_i \in D_{x_i}$. Let $G = (V, E)$ be the connected graph desribing the way in which these processors are linked. The processors proceed by exchanging messages according to some communication protocol, until one of the processors is able to compute f. Formally, a multi–party communication protocol π consists a collection of $r(\pi)$ message functions $m_1(x_1, \ldots, x_n), \ldots, m_{r(\pi)}(x_1, \ldots, x_n)$ satisfying certain information requirements. In particular, let T_{ij} denote the set of indices k such that the k-th message m_k is sent by processor P_i to processor P_j. Then, there exist functions $\hat{m}_1, \ldots, \hat{m}_{r(\pi)}$ such that for each k, $(1 \leq k \leq r(\pi))$, if the k-th

message is sent by processor P_i, then

$$m_k(x_1,\ldots,x_n) = \hat{m}_k\left(x_i, \{m_l(x_1,\ldots,x_n);\ \forall l < k,\ \text{s.t. } l \in T_{ji},\ \text{for some } (j,i) \in E\}\right), \tag{5.1.1}$$

for $k = 1,\ldots,r(\pi)$. In other words, each message must be a function of all the previously received messages and the local input. In light of Eq. (5.1.1), we see that T_{ij} is empty for all $(i,j) \notin E$, which corresponds to the physical requirement that messages can only be sent through the communication channels. In addition to Eq. (5.1.1), the message functions $m_1,\ldots,m_{r(\pi)}$ have to satisfy another requirement, namely, they should provide enough information for one of the processors to evaluate f. Formally, this means that there exists an integer i $(1 \leq i \leq n)$ and a function h such that:

$$f(x_1,\ldots,x_n) = h\left(x_i, \{m_l(x_1,\ldots,x_n); l \in T_{ji},\ \text{for some } (j,i) \in E\}\right), \tag{5.1.2}$$

$$\forall x_1 \in D_{x_1},\ldots,\forall x_n \in D_{x_n}.$$

(This corresponds to the situation where processor P_i performs the final evaluation of f.)

Similar to the way in which the two–way communication complexity is defined, we can introduce smoothness constraints, (such as continuous differentiability or polynomiality), on the functions $\hat{m}_1,\ldots,\hat{m}_{r(\pi)}$, and define for the corresponding class of protocols the multi–party communication complexity of computing f.

Even though Abelson's result is obtained for the case of two–party communication, it can, nonetheless, still be applied to the multi–party case by partitioning the network into two parts and considering each part as a single "processor". However, such an approach is often not effective since Abelson's bound takes into account only the analytic properties of f and does not exploit the network topology in which the processors are connected. Evidently, a good lower bound will have to take into consideration both the analytic properties of f and the combinatorial nature of the graph $G = (V,E)$. But, as we shall see in the next subsection, such a goal may be elusive because the combinatorial issues alone are already computationally intractable.

5.1.2 Computing a Linear Function

Let $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} a_i x_i$, where $x_i \in \Re$ and each a_i is some scalar. Suppose that there are n processors $P_1, \ldots, P_n$ who are connected in a network described by an undirected graph $G = (V, E)$ and wish to compute f, but each processor P_i knows only the values of $a_1, \ldots, a_n$ and x_i.

If all of the coefficients a_i $(i = 1, \ldots, n)$ of f are nonzero, then we claim that the minimum number of messages that have to be exchanged for computing f is $n - 1$. In fact, suppose that processor P_n performs the final computation (cf. Eq. (5.1.2)), then since f depends on variables $x_1, \ldots, x_{n-1}$, it follows that each processor P_i $(1 \leq i \leq n-1)$ has to send out at least one message so that processor P_n can take into consideration of variable x_i. Thus, $r(\pi) \geq n - 1$ for every protocol that computes f. On the other hand, $n - 1$ messages are also sufficient. Let T be a spanning tree of $G = (V, E)$ rooted at node n. For any pair of u, v in T, we say that u is the successor of v, or v is a predecessor of u, if u is on the path from node v to the root node. We say that u is an immediate successor (respectively, predecessor) of v if u is a successor (respectively, predecessor) of v and $\{u, v\}$ is an edge in T. The following protocol π uses $n - 1$ messages to compute f. In the first phase, each processor P_i computes the value of $a_i x_i$. In the second phase, these values are gathered along the spanning tree T in a way such that exactly one message is transmitted over each edge of T. In particular, each processor P_i will compute the sum of all the messages received from its predecessors, add to it the value of $a_i x_i$, and then send the resulting value to its successor. (Each leaf node sends the value $a_i x_i$ only to its predecessor, since it does not have any predecessor.) It is easy to see that such a protocol uses exactly $n - 1$ messages and works correctly. This implies that the communication complexity of computing f with n connected processors is $n - 1$.

For the case where some of the a_i's are zero, the situation is slightly more complicated. Without loss of generality, let us assume that $a_i \neq 0$ for $i = 1, \ldots, m$, and $a_i = 0$ for $i = m+1, \ldots, n$. We use V' to denote the set of nodes $\{1, \ldots, m\}$ of G. An *edge-connecting set* E' for V' is a set of edges in E that connect the nodes of V'. Let K be the minimum cardinality of any edge–connecting set for V'. We claim

that K is equal to the minimum number of messages that has to be exchanged to compute f. For any communication protocol π that computes f, let E' be the set of edges along which some message of π is transmitted. Note that f depends on the variables $x_1, \ldots, x_m$. It follows that the edges of E' connect the nodes in V', because otherwise the values of certain variables cannot affect the final output f and thus π cannot correctly compute f. This implies that $|E'| \geq K$. But $r(\pi) \geq |E'|$, therefore, we have $r(\pi) \geq K$. Conversely, suppose $E' \subset E$ is a collection of K edges that connect the nodes of V'. Let $\overline{V}$ be the set of nodes adjacent to the edges of E'. Then, the graph $G' = (\overline{V}, E')$ is connected and $V' \subset \overline{V}$. Now, the function f can be computed by the processors and communication channels corresponding to G'. By the same argument as before, we can choose some spanning tree T' of G' and accumulate the sum $\sum_{i=1}^{m} a_i x_i$ along the edges of T' in a way such that there is only one message transmitted through each edge. But the edges of T' belong to E' and $|E'| = K$, we see that f can be computed with no more than K edges. This implies that K is equal to the minimum number of messages that have to be exchanged among procssors $P_1, \ldots, P_n$, who are linked in a fashion described by the undirected graph G, in order to compute f.

In summary, the problem of finding an optimal communication protocol to compute f has been reduced to the following *Steiner Tree* problem:

Steiner Tree Problem

Instance: Given a graph $G = (V, E)$, a subset $V' \subset V$ and an integer K $(1 \leq K \leq |E|)$.

Question: Is there an edge–connecting set $E' \subset E$ for V' such that $|E'| \leq K$?

Unfortunately, the Steiner Tree problem has been shown to be NP–Complete (see [GJ 79]). This implies that finding a multi–party communication problem is, in general, combinatorially very difficult, even in the case where f is as simple as a linear function.

5.2 Discrete Multi–Party Protocols With a Tree Structure

In what follows, we let $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i$, where each $x_i \in [0,1]$. We consider a situation in which a collection of processors $P_1, \ldots, P_n$ are to compute the value of f up to a prespecified accuracy ϵ, under the assumption that each processor P_i has knowledge of the value x_i only. The processors are linked via a network of communication channels described by a connected graph $G = (V, E)$. Here, $|V| = n$ and each node i of G corresponds to the processor P_i.

We make the following standing assumptions:

a) Messages can be transmitted only along certain restricted communication channels, namely, those correspond to the arcs of some prespecified spanning tree T of G. Moreover, the messages go only in the direction towards the root of T;

b) The messages sent by each processor P_i are a string of 0–1 bits, and are sent only to the immediate successor of P_i;

c) The length of the 0–1 string sent by each processor P_i is a fixed finite number d_i (i.e., independent of the values of $x_1, \ldots, x_n$.).

d) The root of T performs the final evaluation of f;

Let $p(1), \ldots, p(m)$ be the set of immediate predecessors of some processor P_i. Then, processor P_i receives a message α_l from each $p(l)$. Each message α_l represents, up to a certain degree of accuracy, the value $\sum_{k \in C(p(l))} x_k$. (Here, we use the notation $C(i)$ to denote the set of predecessors of node i.) In other words, processor interprets the message α_l as the indication that the value of $\sum_{k \in C(p(l))} x_k$ lies in a given subset E^l of $[0, |C(p(l))|]$. Thus, after receiving all the messages from its immediate predecessors, processor P_i will have known, up to a certain degree of accuracy, the value $\sum_{k \in C(i)} x_k$. To generate the next message, processor P_i partitions the unit interval $[0,1]$ into 2^{d_i} mutually disjoint sets E_i^j, $j = 1, \ldots, 2^{d_i}$, and sends a binary string of length d_i to its successor according to which set E_i^j the value x_i lies in. The size and the exact way of such partition depends on the values of the messages $\alpha_1, \ldots, \alpha_m$. We make a further assumption:

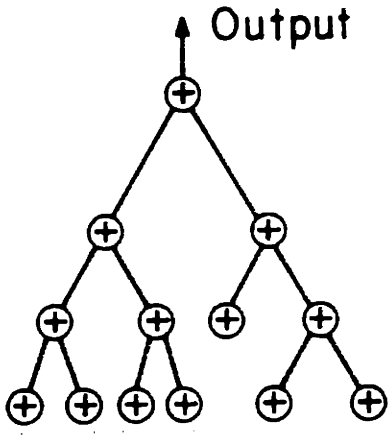

Figure 5.1: Computing $\sum_{i=1}^{13} x_i$ using a protocol with a tree structure.

e) Each set E_i^j is measurable (with respect to Lebesque measure), $j = 1, \ldots, 2^{d_i}$.

The computation of f proceeds in a bottom–up fashion: the processors corresponding to the leaf nodes in T start by sending messages to their immediate successors; each intermediate processor P_i, upon receiving all the messages from its direct predecessors, then determines the range of x_i and $\sum_{k \in C(i)} x_k$ and informs its successor of its finding approximately; finally, the root processor computes the value of f by using the information received from its immediate successors. Figure 5.1 contains a simple example in which 13 processors are to compute $\sum_{i=1}^{13} x_i$ according to a protocol with a tree structure.

Our main result is the following:

Theorem 5.2.1 *Let $\epsilon > 0$. Suppose that processors $P_1, \ldots, P_n$ are to compute $f = \sum_{i=1}^{n} x_i$ ($x_i \in [0,1]$) up to ϵ accuracy, using a protocol with a tree structure described above. Then, at least $(n-1)\log \frac{n-1}{\epsilon}$ bits must be transmitted.*

Proof: Suppose that π is a protocol for computing f which satisfies Assumptions (a)–(e). Let us introduce some notations. For each $i \in V$, we let $\Lambda(i)$ denote the

set of all possible message sequences sent by the nodes of $C(i)$ and node i. For each $\alpha \in \Lambda(i)$, we let $G^{\alpha}(i)$ denote the range of $x_i + \sum_{k \in C(i)} x_k$ over all possible values of the x_k's ($k \in C(i)$) and x_i for which the message squence generated by π is α. By Assumption (e), we see that $G^{\alpha}(i)$ is measurable, for all i. Next, we show by induction on i, that there exists a message sequence $\alpha \in \Lambda(i)$ such that

$$\mu(G^{\alpha}(i)) \geq \frac{1}{2^{d_i}} + \sum_{k \in C(i)} \frac{1}{2^{d_k}}, \tag{5.2.1}$$

where $\mu(\cdot)$ denotes the Lebesque measure.

We first establish the basis of the induction. Let i be some leaf node of T. Notice that $C(i)$ is empty. Thus, $\Lambda(i)$ is consisted of 2^{d_i} binary strings α_l, $l = 1, \ldots, 2^{d_i}$, of length d_i. Notice that the sets $G^{\alpha_l}(i)$, $l = 1, \ldots, 2^{d_i}$, forms a partition of $[0,1]$. Thus,

$$\begin{aligned} \max_l \{\mu(G^{\alpha_l}(i))\} &\geq \frac{1}{2^{d_i}} \sum_{l=1}^{2^{d_i}} \mu(G^{\alpha_l}(i)) \\ &= \frac{1}{2^{d_i}}, \end{aligned}$$

which implies that Eq. (5.2.1) holds for all the leaf nodes of T.

We now show that Eq. (5.2.1) holds for an arbitrary node i of T under the assumption that each node $k \in C(i)$ satisfies the inductive hypothesis (5.2.1). Let $p(1), \ldots, p(m)$ be the direct predecessors of node i, where m is some positive integer. Then, there exists a message sequence $\alpha_k \in \Lambda(p(k))$, $k = 1, \ldots, m$, such that

$$\mu(G^{\alpha_k}(p(k))) \geq \frac{1}{2^{d_{s(k)}}} + \sum_{l \in C(s(k))} \frac{1}{2^{d_l}}. \tag{5.2.2}$$

Upon receiving a message from each of its predecessors $p(1), \ldots, p(m)$, processor P_i will partition $[0,1]$ into 2^{d_i} disjoint measurable sets $E_i^1, \ldots, E_i^{2^{d_i}}$ and send to its successor a different binary string (of length d_i) according to which set E_i^l ($l = 1, \ldots, 2^{d_i}$) the value x_i lies in. Without loss of generality, we assume that $\mu(E_i^1) = \max_l \{\mu(E_i^l)\}$. Let β be the binary string (sent by P_i) that corresponds to the set E_i^1 and let α be the concatenation of the message sequences $\alpha_1, \ldots, \alpha_m, \beta$. Obviously, $\alpha \in \Lambda(i)$. Moreover, we have

$$G^{\alpha}(i) = G^{\alpha_1}(p(1)) \oplus \cdots \oplus G^{\alpha_m}(p(m)) \oplus E_i^1. \tag{5.2.3}$$

We need the following lemma.

Lemma 5.2.1 *Let E_1, E_2 be some bounded measurable sets in $\Re^1$. Let $E_1 \oplus E_2 = \{x + y \mid x \in E_1,\ y \in E_2\}$. Then,*

$$\mu(E_1 \oplus E_2) \geq \mu(E_1) + \mu(E_2), \tag{5.2.4}$$

where $\mu(\cdot)$ denotes the Lebesque measure.

Proof: Since E_1 and E_2 are bounded, we see that $a = \inf_{x \in E_1} x$ and $b = \sup_{x \in E_2} x$ are both finite. We first establish Eq. (5.2.4) under the assumption that $a \in E_1$ and $b \in E_2$. With this assumption, we see that both the set $a \oplus E_2$ and the set $b \oplus E_1$ are subsets of $E_1 \oplus E_2$. Notice that the set $a \oplus E_2$ lies entirely to the left of the set $b \oplus E_1$ on the real line and these two sets overlap at only one point, namely, $a + b$. Thus, we have

$$\begin{aligned} \mu(E_1 \oplus E_2) &\geq \mu\left((a \oplus E_2) \bigcup (b \oplus E_1)\right) \\ &= \mu(a \oplus E_2) + \mu(b \oplus E_1) \\ &= \mu(E_2) + \mu(E_1), \end{aligned}$$

which proves Eq. (5.2.4).

Now suppose that $a \notin E_1$ or $b \notin E_2$. Let $\{a_i \in E_1\}$ and $\{b_i \in E_2\}$ be two sequences of points such that $\lim_{i \to \infty} a_i = a$ and $\lim_{i \to \infty} b_i = b$. For each i, let

$$E_1^i = \{x \mid x \geq a_i, x \in E_1\}, \text{ and } E_2^i = \{x \mid x \leq b_i, x \in E_2\}.$$

Then, we see that $a_i = \inf_{x \in E_1^i} x$ and $b_i = \sup_{x \in E_2^i} x$. In light of what we have just shown, we have

$$\mu(E_1 \oplus E_2) \geq \mu(E_1^i \oplus E_2^i) \geq \mu(E_1^i) + \mu(E_2^i).$$

Therefore, if we take limit of both sides of the above inequality as i goes to infinity, then we have

$$\begin{aligned} \mu(E_1 \oplus E_2) &\geq \lim_{i \to \infty} \mu(E_1^i) + \lim_{i \to \infty} \mu(E_2^i) \\ &= \mu(E_1) + \mu(E_2), \end{aligned}$$

where the last step follows from the continuity property of Lebesque measure. This completes the proof of Lemma 5.2.1. **Q.E.D.**

Remark: Lemma 5.2.1 is very similar to the so called Brunn–Minkowski inequality (see [BL 76]), except that Lemma 5.2.1 assumes the boundedness of E_1 and E_2 instead of the conditions that $\mu(E_1) > 0$, $\mu(E_2) > 0$ which were used in Brunn–Minkowski inequality.

Using Lemma 5.2.1 and Eqs. (5.2.2)–(5.2.3), we obtain

$$\begin{aligned}
\mu(G^{\alpha}(i)) &= \mu\left(G^{\alpha_1}(s(1)) \oplus \cdots \oplus G^{\alpha_m}(s(m)) \oplus E_i^1\right) \\
&\geq \sum_{k=1}^{m} \mu(G^{\alpha_k}(s(k)) + \mu(E_i^1) \\
&\geq \sum_{k=1}^{m} \left(\frac{1}{2^{d_{s(k)}}} + \sum_{l \in C(s(k))} \frac{1}{2^{d_l}} \right) + \frac{1}{2^{d_i}} \\
&= \frac{1}{2^{d_i}} + \sum_{k \in C(i)} \frac{1}{2^{d_k}},
\end{aligned}$$

which completes the proof of Eq. (5.2.1) for node i.

Let the root node of T be denoted by r. In light of Eq. (5.2.1), there exists a message sequence $\alpha^* \in \Lambda(r)$ such that

$$\mu\left(G^{\alpha^*}(r)\right) \geq \sum_{i \in V, i \neq r} \frac{1}{2^{d_i}}.$$

Notice that the set $G^{\alpha^*}(r)$, which is the range of $\sum_{i \neq r} x_i$ over all possible values of x_i's ($i \neq r$, $i \in V$) such that the message sequence generated by π is α^*, has to be contained in a interval of width at most ϵ, because otherwise the root processor P_r would not be able to compute f up to ϵ accuracy over the set $G^{\alpha^*}(r)$. Thus, we have

$$\sum_{i \in V, i \neq r} \frac{1}{2^{d_i}} \leq \mu\left(G^{\alpha^*}(r)\right) \leq \epsilon.$$

By the convexity of the function 2^{-x}, we see that

$$\begin{aligned}
2^{-\frac{\sum_{i \in V, i \neq r} d_i}{n-1}} &\leq \frac{1}{n-1} \sum_{i \in V, i \neq r} \frac{1}{2^{d_i}} \\
&\leq \frac{\epsilon}{n-1},
\end{aligned}$$

which implies that

$$\sum_{i \in V, i \neq r} d_i \geq (n-1) \log \frac{n-1}{\epsilon}.$$

Q.E.D.

Notice that the bound in Theorem 5.2.1 is quite tight. In fact, let us consider the protocol π described by:

(1) each processor P_i replaces x_i with a new number $\bar{x}_i$ such that $|x_i - \bar{x}_i| \leq \frac{\epsilon}{n}$ and $\bar{x}_i$ can be represented in $O(\log \frac{n}{\epsilon})$ bits;

(2) the processors accumulate the sum of $\sum_{i=1}^n \bar{x}_i$ through the spanning tree T in the same manner as described in Section 5.1.

Since $|f - \sum_{i=1}^n \bar{x}_i| \leq \epsilon$, the protocol π guarantees that the root processor can compute f within ϵ accuracy. Moreover, after a simple counting, we see that the total number of bits transmitted in the protocol π is at most $O(n \log \frac{n}{\epsilon})$, which comes within only a constant factor to the lower bound in Theorem 5.2.1.

Finally, we consider the problem of computing $f(x_1, \ldots, x_n) = \prod_{i=1}^n x_i$, where each $x_i \in [1, 2]$, up to certain prespecified accuracy ϵ with protocols defined before. Using the transformation $y_i = \log x_i$, $i = 1, \ldots, n$, it is not hard to see that this problem is equivalent to the problem of computing $g(y_1, \ldots, y_n) = \sum_{i=1}^n y_i$ to the degree of accuracy $O(\epsilon)$. Thus, as a corollary of Theorem 5.2.1, we have

Corollary 5.2.1 *Let $\epsilon > 0$. Suppose that processors $P_1, \ldots, P_n$ are to compute $f = \prod_{i=1}^n x_i$ ($x_i \in [1, 2]$) upto ϵ accuracy using protocols with a tree structure described before. Then, at least $\Omega\left((n-1) \log \frac{n-1}{\epsilon}\right)$ bits must be transmitted. Moreover, this lower bound is tight in the sense that it can be attained by some protocol with a tree structure.*

5.3 Discussion and Possible Extensions

The field of multi–party communication is basically undeveloped. This is partly because it is hard to formulate a problem in this field so that a theoretical study of its communication complexity is possible. In this chapter, we have shown some

partial results about multi–party communication. Unlike the two–party communication case where the main difficulty lies in studying the analytical properties of f (the function to be computed), new issues, such as the topological structure of the network in which the processors are connected, have to be taken into consideration in the area of multi–party communication. Our approach is to isolate the combinatorial issues by assuming that f has a extremely simple analytical structure (e.g., linear functions). Two kinds of communication protocols have been considered, namely, the continuous protocols and discrete protocols.

In continuous communication protocols, the messages are assumed to be real–valued functions. For this class of protocols, we have considered the problem where a set of processors $P_1, \ldots, P_n$, who are connected in a fashion described by an undirected graph G, are to compute a linear function $f = \sum_{i=1}^{n} a_i x_i$. We assume that each processor P_i has access only to the value of variable x_i $(i = 1, \ldots, n)$. We have shown that the problem of designing an optimal multi–party continuous protocol for computing f, in the case where some of the a_i's are zero, is reduced to the problem of finding a minimum edge–connecting set (MECS) for the graph G. We then transformed the 3–Satisfiability problem to the MECS problem and proved that the minimum edge–connecting set problem is NP–Complete. This result reflects that the combinatorial issues in the area of multi–party communication can be computationally intractable.

We have also considered discrete multi–party communication protocols in which messages are assumed to be binary strings of finite length. We studied the same problem where a set of processors are to compute a linear function $f = \sum_{i=1}^{n} x_i$ with a discrete multi–party protocol, under the same assumptions as before except that: (1) f is computed only within ϵ accuracy; (2) the messages can be sent only along a given set of channels which correspond to a spanning tree in G. We have derived almost optimal upper and lower bounds for this problem. Similar bounds were also obtained for the case where $f = \prod_{i=1}^{n} x_i$, and each x_i lies in the interval $[1, 2]$.

These results are very partial and are mainly intended to illustrate the new issues that arise in the context of multi–party protocols. In the rest of this setion, we describe several interesting problems that are worth studying. In Section 5.1, we have identified one problem, namely, computing a linear function, which is com-

pletely trivial for the case of two–party continuous communication but becomes quite hard for the multi–party case. It is of interest to find more of such problems so as to illustrate the new difficulties introduced to the multi–party communication by the combinatorial structure of graph G. For example, one may consider the problem where f is a quadratic function. (This problem is well solved for the case of two–party communication.) To make the combinatorial aspect of this problem less difficult, one may wish to assume that G is a complete graph.

Another approach in the study of multi–party communication is to isolate the analytic aspect of the problem by making the combinatorial issues easier. In particular, we can assume that G has a special structure. For example, we may consider the problems where G is a complete graph, a tree or a mesh. The result in Section 5.2 can be viewed as a step in this direction. Another way of simplifying the combinatorial issues is to assume that there are a fixed number of processors. For example, one may consider the case where G contains only 3 or 4 processors which are completely connected.

Chapter 6

Conclusions and Future Research Directions

In this thesis, we have considered a typical situation in distributed computation where a set of processors are to collectively perform some computational task, under the assumption that each processor has only partial information about the input data. The processors are connected by a set of communication channels through which messages are exchanged during the course of computation. In this thesis, we have addressed one of the major issues for this problem, namely, the communication requirement among the processors taking part in the computation. In particular, we studied the minimum amount of information (i.e., communication complexity) that has to be exchanged among the processors in order for them to successfully carry out the computation.

In measuring the amount of information exchange, we have used two different kinds of complexity measures: continuous communication model and discrete communication model. In the continuous communication model, messages are assumed to be real–valued (or, complex–valued) and the amount of communication is measured by the total number of real–valued (or, complex–valued) functions that are exchanged. In the discrete communication model, messages are assumed to be binary strings of finite length and the amount of information exchange is defined as the total length of binary strings that are transmitted.

For the most part of this thesis, we have concentrated on the two–party communication case where two processors P_1, P_2 are to evaluate a function $f(x, y)$, under

the assumption that processor P_1 (respectively, P_2) knows only the value of x (respectively, y) and the functional form of f. Messages can either be sent only from processor P_1 to processor P_2 (one–way case), or can be sent in both way (two–way case).

We have presented a variety of new results on the one–way and two–way communication complexity for algebraic problems. We have used, in several occasions, the results of [A 78] and [A 80], but our results are often stronger because they exploit the algebraic structure of the problem. In particular, we have studied (Chapter 2) the one–way communication complexity of computing a set $f_1, \ldots, f_s$ of polynomials. We have extended the results of [A 78] to the case of $s > 1$. We have shown that we can restrict to the class of polynomial protocols while increasing the communication complexity by at most one. We then considered the specia case where $m = n$ and each one of the polynomials $f_i : \Re^n \times \Re^n$ is of the form $f_i(x, y) = \hat{f}_i(x + y)$, where each $\hat{f}_i$ is a polynomial in n variables. For this case, we have given a proof that linear protocols are optimal, and a constructive procedure for designing such protocols. In Chapter 3, we have derived several general lower bounds on the two–way communication complexity of computing a rational function f, using protocols in which messages are constrained to be rational functions of the data. Our results are obtained by combining an earlier result of Abelson [A 80] with Hilbert's Nullstellensatz from algebraic geometry. As an application, we have considered the problem of computing a particular entry of the inverse of $x+y$, where x and y are $n \times n$ complex matrices, and obtained an $n^2 - 1$ lower bound (which agrees with the obvious upper bound, within one message), as opposed to the $\Omega(n)$ lower bound obtained by applying Abelson's result [A 80]. In addition, we have obtained a new lower bound for the two–way continuous communication complexity. Our lower bound is different from that of Abelson's ([A 80]) in that it has to do only with the first order derivatives of f instead of the second order of derivatives used by [A 80]. We then applied this result to the problem where f is defined as a root z of the polynomial $\sum_{i=0}^{n-1}(x_i + y_i)z^i$ obtained a lower bound n. This is in contrast to the $\Omega(1)$ lower bound obtained by applying Abelson's result. We have also considered the problem of computing multiple functions and have examined some of the related combinatorial issues. For the special case where the functions to be computed are simple quadratic polynomials, we have transformed

the problem of designing optimal protocols into the problem of finding a minimum node cover for certain bipartite graph, which is solvable in polynomial time using bipartite matching algorithms.

Chapter 4 is devoted to the study of discrete communication complexity. We have pointed out that Abelson's lower bound for the two–way continuous protocols reamains valid for discrete protocols, provided that we view each Boolean function as a polynomial over an appropriately defined field. Then, we considered a situation where two processors P_1 and P_2 are to approximately minimize the sum of two convex functions $f_1 + f_2$, under the assumption that each processor P_i has access to the function f_i $(i = 1, 2)$ only. We have provided both upper and lower bounds on the number of bits that have to be exchanged between the two processors in order to find an ϵ–optimal solution. We then applied these results and a result of Dudley ([D 74]) to the problem of deciding whether two given convex sets are (almost) disjoint and show that two–way communication protocols can provide exponential savings over one–way communication protocols. (This result was first proved by Papadimitriou and Sipser [PS 82].)

We have also given some preliminary results for the case of multi–party communication (Chapter 5). Both continuous and discrete communication protocols are considered. We have illustrated some new issues that arise in the context of multi–party communication by demonstrating that the problem of designing optimal continuous multi–party protocols for computing simple functions (such as the linear functions) is combinatorially intractable. Then, we considered a class of multi–party protocols in which the communication channels have a tree structure and the messages are binary strings. For such class of protocols, we show an almost tight lower bounds on the communication complexity of computing simple functions such as $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i$, or $f(x_1, \ldots, x_n) = \prod_{i=1}^{n} x_i$, up to some prespecified accuracy ϵ.

Before we close this thesis, we suggest several general directions for future reseach. First of all, we believe that some of the results obtained in thesis can be improved. For example, new lower bounds for two–way communication complexity may be obtained by taking into consideration the higher order derivatives of f. Another potential area of future research concerns two–way continuous protocols

for computing a collection of functions $\{f_1, \ldots, f_s\}$, with $s > 1$. Here, even if one assumes that the functions f_i are quadratic, the evaluation of the communication complexity is surprisingly hard and leads to problems with a combinatorial flavor. Some partial results can be found in Chapter 3. Finally, we believe that more applications, from areas such as system theory, data communication and digital signal processing, of the existing results should be considered.

There is another important research direction which is worth pursuing. In this thesis, all the communication channels are assumed to be noiseless and reliable. But, in many practical applications, one is often faced with the reality that the messages transmitted among the processors may be subject to distortion. We can formulate this problem as follows. For simplicity, let there be two processors P_1, P_2 who are to compute a Boolean function $f(x, y)$. As usual, we assume that processor P_1 (respectively, P_2) has access to the value of Boolean variable $x \in \{0,1\}^m$ (respectively, $y \in \{0,1\}^n$). The messages are assumed to be binary strings and the processors communicate in the same manner as described in Subsection 1.1.1 of chapter 1, except that the mesages may be corrupted by noise. The presence of noise is modelled by assuming that a certain (fixed) number of errors may occur in the course of informatin transmission. More specifically, let K be an integer (independent of n). We assume that there can be at most K transmission errors in the course of communication. (An error of transmission means that the bit 0 (or 1) is received when actually the bit 1 (respectively, 0) is sent.) The exact places where these errors can occur are assumed to be arbitrary. We are interested in the minimum number of bits (i.e., the communication complexity) that have to be exchanged in order to correctly compute f, as a function of n and K. Notice that the communication complexity as defined in this context, is always finite. This is because the processors can correctly simulate a deterministic protocol in the following way: each processor can always send each message repeatedly for $2K + 1$ times so as to ensure the other processor can decide on the value of this message correctly, and this implies that a finite amount of communication is sufficient for computing f in the presence of a fixed number transmission errors. (There is a slight variation of this problem, that is, instead of constraining the total number of transmission errors that is allowed to occur in both directions, we constrain the number of the allowable transmission errors in each direction of communication

separately.) Interestingly, the one–way version of this problem (i.e., the messages are sent only from processor P_1 to processor P_2) has been studied by other researchers before, but in a different context. In particular, the following problem was stated by S.M. Ulam [U 76, page 281] in this way:

> Someone thinks of a number between one and one million (which is just less than 2^{20}). Another person is allowed to ask up to twenty questions, to which the first person is supposed to answer only yes or no. Obviously, the number can be guessed by asking first: Is the number in the first half–million? and then again reduce the reservior of numbers in the next question by one–half, and so on. Finally the number is obtained in less than $\log_2(1000000)$. Now suppose one were allowed to lie once or twice, then how many questions would one need to get the right answer?

It turns out that the answer is 25 ([N 88], [S 84]). [N 88] also contains a general solution to this problem (where only one is allowed to lie only once). The solution is based on the error correcting code theory ([MS 77]). In a similar framework, [KM 80] studied the problem of performing binary search in the presence of errors. If we imagine the two people as two processors and the lies as errors occured during communication, then Ulam's problem can be casted in the framwork of one–way communication in the presence noise. We believe that more work is needed in this direction, especially for the case of two–way communication. The techniques from the theory of error–correcting code may be applicable in this context.

Finally, the area of multi–party communication is largely open and much future research is needed. The results obtained in Chapter 5 are partial and many important issues in this area remain unresolved. For example, for the continuous model of communication, there has been no general lower bound approach for studying the multi–party communication complexity which takes into account both the analytical properties of f and the combinatorial structure of the graph G. (Here, f is the function to be computed and G is the graph describing the network structure in which the processors are connected.) Of course, one can select an arbitrary partition of graph G, apply Abelson's result to obtain a lower bound for the minimum amount of information exchange between the two sets of processors in the partition, and

combine these lower bounds for various partitions to obtain a bound on the total amount of communication among the processors. While this approach has enjoyed limited success ([A 80]), we believe that it does not fully exploit the topological structure of G. We believe that an effective general lower bound approach must take into more careful consideration of the network structure of G. Similar to the continuous multi–party communication, the discrete multi–party communication is also very poorly understood. The main difficulty lies in the problem formulation. The problems in this area, when formulated in the most general form, are often combinatorially too difficult to analyze. It seems that certain assumptions on the communication protocols are neccessary in order to draw some nontrivial conclusions. For example, the work of [CF 83] is based on a special assumption on the way in which information is exchanged among the processors. It would be interesting to find some other assumptions on communication protocols, under which nontrivial lower bounds can be obtained.

Appendix A

Multi–Variable Calculus

This appendix contains some results concerning multi–variable functions that are used in Chapter 3.

Let $F : U \times V \mapsto \Re^s$ be a continuously differentiable mapping, where U and V are open subsets of $\Re^r$ and $\Re^t$ respectively. We assume that $r > s$. Let $(u^*, v^*) \in U \times V$ be such that $\text{rank}[\nabla_u F(u^*, v^*)] = s$. Then, the matrix $\nabla F_u(u^*, v^*)$ has s linearly independent rows and we can find a set $J \subset \{1, \ldots, r\}$ of indices, of cardinality s, such that the vectors $\left(\frac{\partial F_1}{\partial u_i}(u^*, v^*), \ldots, \frac{\partial F_s}{\partial u_i}(u^*, v^*)\right)$, $i \in J$ are linearly independent. We define the projection $\Pi : \Re^r \mapsto \Re^{r-s}$ by letting $\Pi(u)$ be the vector with coordinates u_i, $i \notin J$. We have the following lemma.

Lemma A.1.0 *There exists a connected open subset R of $U \times V$, and a connected open set $S \subset \Re^{r+t}$, and a continuously differentiable function $g : S \mapsto R$ such that $(u^*, v^*) \in R$,*

$$S = \{ \, (F(u,v), \Pi(u), v) \mid (u,v) \in R \, \},$$

and such that

$$z = g\left(F(u,v), \Pi(u), v\right), \quad \forall (u,v) \in R. \tag{A.1.1}$$

Proof: Consider the mapping $q : U \times V \mapsto \Re^{r+t}$ defined by $q(u,v) = (F(u,v), \Pi(u), v)$. We claim that $\nabla q(u^*, v^*)$ has full rank. To see this, let us permute the rows of $\nabla q(u^*, v^*)$ so that the last $r + t - s$ rows correspond to the partial derivatives with

respect to the variables v and u_i, $i \notin J$. Then, $\nabla q(u^*, v^*)$ will have the structure

$$\nabla q(u^*, v^*) = \begin{bmatrix} A & 0 \\ B & I \end{bmatrix},$$

where A, B are suitable submatrices of $\nabla F(u^*, v^*)$ and I is the $(r + t - s) \times (r + t - s)$ identity matrix. Each one of the s rows of matrix A is a vector of the form $\left(\frac{\partial F_1}{\partial u_i}(u^*, v^*), \ldots, \frac{\partial F_s}{\partial u_i}(u^*, v^*)\right)$, $i \in J$, and these vectors are linearly independent by construction. Thus $\det(\nabla q(u^*, v^*)) = \det(A) \neq 0$. The result then follows from the inverse function theorem [S 65, page 35]. **Q.E.D.**

Theorem A.1.0 *Let Q be an open subset of $\Re^r$. Let $F : Q \mapsto \Re^s$ be a continuously differentiable mapping such that*

$$\max_{z \in Q} \operatorname{rank} (\nabla F(z)) = s. \tag{A.1.2}$$

Suppose that $f : Q \mapsto \Re$ is a continuously differentiable function with the property

$$\nabla f(z) \in \operatorname{span} \{\nabla F(z)\}, \qquad \forall z \in Q.$$

Then, there exists some continuously differentiable function h such that $f(z) = h(F(z))$ for all $z \in R$, where R is some open subset of Q.

Proof: Suppose that $z^* \in Q$ is a vector at which the maximum in Eq. (A.1.2) is attained. By taking $t = 0$ and dropping the set V, we see that all the assumptions of Lemma A.1.0 are satisfied[1], and thus Lemma A.1.0 applies. Let R, S and g be as in Lemma A.1.0. By assumption, $\nabla f(z) \in \operatorname{span} \{\nabla F(z)\}$, $\forall z \in R$. Thus, for every $z \in R$, there exists a vector $d(z) \in \Re^s$ such that

$$\nabla f(z) = \nabla F(z) d(z), \quad \forall z \in R. \tag{A.1.3}$$

Using Lemma A.1.0, we have

$$F(z) = F\left(g(F(z), \Pi(z))\right), \quad \forall z \in R,$$

[1] We have assumed that $r > s$ here. The proof for the case $r = s$ is essentially the same except that Π is redundant.

or

$$u = F(g(u,v)), \quad \forall (u,v) \in S. \tag{A.1.4}$$

Let $\nabla_v g$ be the $(r-s) \times r$ matrix of the partial derivatives of g, with respect to the components of v. Since the left hand side of Eq. (A.1.4) does not depend on v, the chain rule yields

$$0 = \nabla_v g(u,v) \cdot \nabla F(g(u,v)), \quad \forall (u,v) \in S. \tag{A.1.5}$$

We use Lemma A.1.0 once more to obtain

$$f(z) = f\left(g(F(z), \Pi(z))\right), \quad \forall z \in R.$$

We define a function $\bar{h} : S \mapsto \Re$ by letting

$$\bar{h}(u,v) = f(g(u,v)), \quad \forall (u,v) \in S. \tag{A.1.6}$$

Notice that $\bar{h}$ is continuously differentiable. Using the chain rule,

$$\nabla_v \bar{h}(u,v) = \nabla_v g(u,v) \cdot \nabla f(g(u,v)), \quad \forall (u,v) \in S,$$

where $\nabla_v \bar{h}(u,v)$ is the vector of partial derivatives of $\bar{h}$ with respect to the components of v. Using (A.1.3) and (A.1.5), we conclude that $\nabla_v \bar{h}(u,v) = 0$, for all $(u,v) \in S$. Since S is open and connected, it is easily shown that $\bar{h}$ is independent of v and there exists a function $h : V \mapsto \Re$ such that

$$\bar{h}(u,v) = h(u), \quad \forall (u,v) \in S.$$

Here $V = F(R)$ which is obviously open and connected. For any $z \in R$, we have

$$f(z) = f\left(g(F(z), \Pi(z))\right) = \bar{h}\left(F(z), \Pi(z)\right) = h\left(F(z)\right),$$

as desired. **Q.E.D.**

Theorem A.1.1 *Let $F : Q \mapsto \Re^s$ be continuously differentiable, where $Q \subset \Re^r$ is open. We assume that* $\operatorname{rank}(\nabla F(z)) < s$, $\forall z \in Q$, *and that $\nabla F_1(z)$ (F_1 is the first component mapping of F) is not equal to zero on the set Q. Then, there exists some positive integer i and some continuously differentiable function g such that*

$$F_{i+1}(z) = g\left(F_1(z), \ldots, F_i(z)\right), \qquad \forall z \in R,$$

where R is some nonempty open subset of Q and F_i denotes the i-th component mapping of F.

Proof: We let i be the largest index such that there exists some $\hat{z} \in Q$ with the property

$$\dim \text{span}\{\nabla F_1(\hat{z}), \ldots, \nabla F_i(\hat{z})\} = i.$$

Clearly, $1 \leq i < s$. By continuity, there exists some open subset $\hat{Q}$ of Q containing $\hat{z}$ such that $\nabla F_1(z), \ldots, \nabla F_i(z)$ are linearly independent for all $z \in \hat{Q}$. By our choice of the index i, we have

$$\nabla F_{i+1}(z) \in \text{span}\{\nabla F_1(z), \ldots, \nabla F_i(z)\}, \qquad \forall z \in \hat{Q}.$$

By Theorem A.1.0, we see that there exists a continuously differentiable function $h : U \mapsto \Re$ such that

$$F_{i+1}(z) = h(F_1(z), \ldots, F_i(z)), \qquad \forall z \in R$$

where R is some open subset of $\hat{Q}$ and $U = F(R)$. **Q.E.D.**

Theorem A.1.2 *Let $F : U \times V \mapsto \Re^s$ be a continuously differentiable mapping, where U and V are open subsets of $\Re^r$ and $\Re^t$ respectively. Let $(u^*, v^*) \in U \times V$ be such that $\text{rank}[\nabla_u F(u^*, v^*)] = s$ and $F(u^*, v^*) = 0$. Then, there exists some nonempty open subsets $\overline{V} \subset V$, $W \subset U$ such that $u^* \in W$, $v^* \in \overline{V}$ and*

$$\{\ u \mid F(u, v) = 0\ \} \bigcap W$$

is nonempty and connected for all $v \in \overline{V}$. Furthermore, if $\{v_i \in V; i = 1, 2, \ldots\}$ is a sequence of vectors such that $\lim_{i \to \infty} v_i = v$ and $v \in \overline{V}$, then there exists a sequence $\{u_i \in W\}$ such that $F(u_i, v_i) = 0$ and $\lim_{i \to \infty} u_i = u$ for some $u \in W$.

Proof: We are in a situation where the assumptions of Lemma A.1.0 hold[2]. Let q, g, R, S be given as in Lemma A.1.0. Thus, $(u, v) = g(q(u, v)) = g(F(u, v), \Pi(u), v)$, for all $(u, v) \in R$. Let g_u, g_v be the corresponding component mappings of g such that $u = g_u(q(u, v))$ and $v = g_v(q(u, v))$. Since S is open, we can take an connected open subset of S with the form $W_1 \times W_2 \times \overline{V}$ such that $W_1 \subset \Re^s$, $W_2 \subset \Re^{r-s}$ and $q(u^*, v^*) \in W_1 \times W_2 \times \overline{V}$. It is easy to check that W_2 is nonempty and connected

[2]Here we have assumed that $r > s$. The same argument works for the case $r = s$ except that Π should be dropped in the remaining proof.

and that $v^* \in \overline{V}$. Since g is a diffeomorphism, it follows that the set $g(W_1 \times W_2 \times \overline{V})$ is open. Moreover, we claim that g has following properties:

(1) $g_v(w_1, w_2, v) = v$, for all $(w_1, w_2, v) \in W_1 \times W_2 \times \overline{V}$;

(2) $\Pi(g_u(w_1, w_2, v)) = w_2$, for all $(w_1, w_2, v) \in W_1 \times W_2 \times \overline{V}$.

To prove the first property, let us write $(w_1, w_2, v) = q(u, v')$ for some $(u, v') \in R$. This is possible since $(w_1, w_2, v) \in S$. Hence, $(w_1, w_2, v) = (F(u, v'), \Pi(u), v')$. It follows that $v = v'$ and $(w_1, w_2, v) = q(u, v)$. Thus, $g_v(w_1, w_2, v) = g_v(q(u, v)) = v$, which proves (1). We now show the second property. As we have just seen, there exists some u such that $(w_1, w_2, v) = q(u, v)$ and $(u, v) \in R$. Thus, $(w_1, w_2, v) = (F(u, v), \Pi(u), v)$, from which follows that $w_2 = \Pi(u)$. On the other hand, we have

$$\Pi(g_u(w_1, w_2, v)) = \Pi(g_u(q(u, v))) = \Pi(u),$$

from which follows that $w_2 = \Pi(g_u(w_1, w_2, v))$.

Now let $W = g_u(W_1 \times W_2 \times \overline{V})$ and $S_u(v) = \{\ u \in U \mid F(u, v) = 0\ \}$. Since W is the projection of the open set $g(W_1 \times W_2 \times \overline{V})$, it follows that W is open in $\Re^r$. Also, it can be easily seen that $W \subset U$ and $u^* \in W$. Furthermore, we claim that

$$S_u(v) \bigcap W = \{\ g_u(0, w_2, v) \mid w_2 \in W_2\ \}, \forall v \in \overline{V}. \tag{A.1.7}$$

In fact, let us fix some $v \in \overline{V}$ and let $E(v)$ be the set in the right–hand side of Eq. (A.1.7). We will show that $E(v) \subset S_u(v) \bigcap W$. Clearly, $E(v) \subset W$. Thus, we only need to show that $E(v) \subset S_u(v)$. Let u be an element of $E(v)$. Then, there exists some $w_2 \in W_2$ such that $u = g_u(0, w_2, v)$. Since $q(u^*, v^*) = (F(u^*, v^*), \Pi(u^*), v^*) = (0, \Pi(u^*), v^*)$ and $q(u^*, v^*) \in W_1 \times W_2 \times \overline{V}$, we see that $0 \in W_1$. Thus, $(0, w_2, v) \in W_1 \times W_2 \times \overline{V}$. In light of property (1), we see that $v = g_v(0, w_2, v)$. Consequently,

$$F(u, v) = F\left(g_u(0, w_2, v), g_v(0, w_2, v)\right) = F(g(0, w_2, v)) = 0.$$

It follows that $E(v) \subset S_u(v) \bigcap W$.

For the reverse inclusion, given any $u \in S_u(v) \bigcap W$, we have $F(u, v) = 0$. Furthermore, there exists some $(w_1, w_2, v') \in W_1 \times W_2 \times \overline{V}$ such that $u = g_u(w_1, w_2, v')$. By property (2), we see that $\Pi(u) = w_2$. Thus, $(0, w_2, v) = (F(u, v), \Pi(u), v) = q(u, v)$. Hence, $u = g_u(q(u, v)) = g_u(0, w_2, v)$. This implies that $u \in E(v)$, and Eq.

(A.1.7) has been established. As a result, the set $S_u(v) \bigcap W$ is connected because, according to (A.1.7), it is the image of the connected set W_2 under a continuous mapping. Since $E(v)$ is nonempty for each $v \in \overline{V}$, Eq. (A.1.7) also shows that $S_u(v) \bigcap W$ is nonempty.

Given a sequence of vectors $\{v_i \in \overline{V}; i = 1, 2, \ldots\}$ such that $\lim_{i \to \infty} v_i = v$ and $v \in \overline{V}$, let us pick $u_i = g_u(0, w_2, v_i)$, $i = 1, 2, \ldots$, where w_2 is some fixed vector in W. Hence, $u_i \in E(v_i)$, for all i. According to Eq. (A.1.7), we see that $F(u_i, v_i) = 0$. Furthermore, by the continuity of g_u, we see that

$$\lim_{i \to \infty} u_i = \lim_{i \to \infty} g_u(0, w_2, v_i) = g_u(0, w_2, v),$$

which is clearly in W. **Q.E.D.**

Theorem A.1.3 *Let Q be an open set in $\Re^t$. Let also $F : Q \mapsto \Re^s$ be a continuously differentiable mapping such that*

$$\text{rank}\,(\nabla F(z)) = s, \qquad \forall z \in A, \tag{A.1.8}$$

where $A = \{\ z \mid F(z) = 0\ \}$. Suppose that $f : Q \mapsto \Re$ is continuously differentiable and is a constant function of z on A. Then,

$$\nabla f(z) \in \text{span}\,\{\nabla F(z)\}, \qquad \forall z \in A. \tag{A.1.9}$$

Proof: Consider the following constrained optimization problem:

$$\min_{z \in A} f(z). \tag{A.1.10}$$

By assumption, each z in A is an optimal solution to (A.1.10). Since the regularity condition (Eq. (A.1.8)) ensures the existence of a set of Lagrange multipliers, the necessary condition for optimality gives the desired result ([L 84, page 300]). **Q.E.D.**

References

[A 78] Abelson, H., " Towards a Theory of Local and Global Computation", *Theoretical Computer Science,* Vol. 6, 1978, pp. 41–67.

[A 80] Abelson, H., "Lower Bounds on Information Transfer in Distributed Computations", *Journal of ACM,* 27, 2, 1980, pp. 384–392.

[AM 69] Atiyah, M. and Macdonald, I., *Introduction to Commutative Algebra*, Addison–Wesley Publishing Company, 1969.

[AU 83] Aho, A.V., Ullman, J.D. and Yannakakis, M., "On Notions of Information Transfer in VLSI Circuits", *Proceedings of the 15th STOC*, 1983, pp. 133–139.

[BL 76] Brascamp, H.J. and Lieb, E.H., "Best Constants in Young's Inequality, Its Converse and Its Generalization to More Than Three Functions", *Advances in Mathematics*, 20, 1976, pp. 151–173.

[BM 75] Borodin, A. and Munro, I., *The Computational Complexity of Algebraic and Numeric Problems,* American Elsevier, New York, 1975.

[BT 89] Bertsekas, D.P. and Tsitsiklis, J.N., *Parallel and Distributed Computation: Numerical Methods*, Prentice Hall, 1989.

[C 76] Csanky, L., "Fast Parallel Matrix Inversion Algorithms", *SIAM Journal on Computing,* Vol. 5, 1976, pp. 618–623.

[CF 83] Chandra, A.K., Furst, M.L. and Lipton, R.J., "Multi–Party Protocols", *Proceedings of the 15th STOC.*, 1983, pp. 94–99.

[D 74] Dudley, R.M., "Metric Entropy of Some Classes of Sets with Differentiable Boundaries", *Journal of Approximation Theory*, Vol. 10, No. 3, March 1974, pp. 227–236.

[DGS 84] Duris, P., Galil, Z. and Schnitger, G., "Lower Bounds on Communication Complexity", *Proceedings of the 16th STOC*, 1984, pp. 81–91.

[F 87] Fürer, M., "The Power of Randomness for Communication Complexity", *Proceedings of 19th STOC*, 1987, pp. 178–181.

[GJ 79] Garey, M.R. and Johnson, D.S., *Computers and Intractability: a Guide to the Theory of NP–Completeness,*, W.H. Freeman and Company, San Froncisco, 1979.

[GV 83] Golub, G.H. and Van Loan, C.H., *Matrix Computations*, Johns Hopkiins Univ. Press, 1983.

[GN 76] Gordan, P. and Nöether, M., "Ueber die Algebraischen Formen, deren Hesse'sche Determinante Identisch Verschwindet", *Math. Ann.*, Vol. 10, 1876, pp. 547–568.

[H 77] Hartshorne, R., *Algebraic Geometry*, Springer–Verlag, New York, 1977.

[HR 88] Halstenberg, B. and Reischuk, R., " On Different Modes of Communication", *Proceedings of 20th STOC*, 1988, pp. 162–172.

[HMT 88] Hajnal, A., Maass, W. and Turan, G., "On the Communication Complexity of Graph Properties", *Proceedings of 20th STOC*, 1988, pp. 186–191.

[J 34] Ja'Ja, J., " The VLSI Complexity of Selected Graph Problems", *Journal of ACM*, Vol. 31, No. 2, 194, pp. 377–391.

[JK 84] Ja'Ja, J. and Kumar, V.K., "Information Transfer in Distributed Computing With Applications to VLSI", *Journal of ACM,* Vol. 31, No. 1, 1984, pp. 150–162.

[JKS 84] Ja'Ja, J., Kumar, V.K. and Simon, J., "Information Transfer Under Different Sets of Protocols", *SIAM Journal of Computing,* Vol. 13, No. 4, 1984, pp. 840–849.

[K 79] Kachian, L.G., "A Polynomial Algorithm for Linear Programming", *Soviet Math. Doklady*, 20, 1979, pp. 191–194.

[KM 80] Kleitman, D.J., Meyer, A.R., Rivest, R.L., Spencer, J. and Winklmann, K., "Coping with Errors in Binary Search Procedures", *Journal of Comput. System Sci.*, 20, 1980, pp. 396–404.

[K 84] Kleiman, S.L., "Tangency and Duality", *Proc. CMS Summer Institute in Algebraic Geometry*, Vancouver, 1984.

[LS] Lipton, R.J. and Sedgewick, R., "Lower Bounds for VLSI", *Proceedings of 13th STOC*, 1981, pp. 300–307.

[L 66] Lorentz, G.G., *Approximation of Functions,* Holt, Rinehart and Winston, 1966.

[L 84] Luenberger, D.G., *Linear and Nonlinear Programming*, Addison–Wesley Publishing Company, 1984.

[LT 89] Luo, Z.Q. and Tsitsiklis, J.N., "On the Communication Complexity of Distributed Algebraic Computaion", technical report LIDS–P–1851, Lab. for Information and Decision Systems, MIT, Cambridge, Mass., submitted to the *Journal of ACM*, 1989.

[LTs 89] Luo, Z.Q. and Tsitsiklis, J.N., "On the Communication Complexity of Solving a Polynomial Equation", technical report LIDS–P–1876, Lab. for Information and Decision Systems, MIT, Cambridge, Mass., submitted to *SIAM Journal on Computing*, 1989.

[MS 77] MacWilliams, F.J. and Sloane, N.J., *The Theory of Error–Correcting Codes*, North–Holland Pub. Co., 1977.

[M 75] Munkres, J.R., *Topology, a first course,* Prentice–Hall, Inc., 1975.

[MS 82] Mehlhorn, K. and Schmidt, E.M., "Las Vegas is Better than Determinism in VLSI and Distributed Computing", *Proceedings of the 14th STOC,* 1982, pp. 330–337.

[M 71] Miranker, M.L., "A Survey Of Parallelism in Numerical Analysis", *SIAM Review,* Vol. 13, No. 4, 1971, pp. 524–547.

[M 86] Mulmuley, K., "A Fast Parallel Algorithm to Compute the Rank of a Matrix Over an Arbitrary Field", *Proceedings of the 18th STOC,* pp. 338–339, 1986.

[NY 83] Nemirovsky, A.S., and Yudin, D.B., *Problem Complexity and Method Efficiency in Optimization,* Wiley, 1983.

[N 88] Niven, I., "Coding Theory Applied to a Problem of Ulam", *Mathematics Magazine*, Vol. 61, 1988, pp. 275–281.

[PE 86] Pang, K.F. and El Gamal, A., "Communication Complexity of Computing the Hamming Distance", *SIAM Journal on Computing*, 15,4, 1986, pp. 932–947.

[PSi 84] Paturi, R. and Simon, J., "Probabilistic Communication Complexity", *Proceedings of the 25th FOCS,* 1984, pp. 118–126.

[PS 82] Papadimitriou, C.H. and Sipser, M., "Communication Complexity", *Proceedings of the 14th STOC,* 1982, pp. 196–200.

[PS 83] Papadimitriou, C.H. and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity,* Prentice Hall, 1982.

[PT 82] Papadimitriou, C.H. and Tsitsiklis, J.N., "On the Complexity of Designing Distributed Protocols", *Information and Control*, 53, 3, 1982, pp. 211-218.

[PU 84] Papadimitriou, C.H. and Ullman, J.D., " A Communication–Time Tradeoff", *Proceedings of 16th STOC*, 1984, pp. 84–88.

[PY 88] Papadimitriou, C.H. and Yannakakis, M., "Towards an Architecture – Independent Analysis of Parallel Algorithms", *Proceedings of 20th STOC*, 1988, pp. 510–513.

[R 76] Rabin, M.O., "Probabilistic Algorithms", *Algorithms and Complexity: Recent Results and New Directions*, Edited by Traub, J.F., Academic Press, 1976, pp. 21–40.

[Ru 76] Rudin, W., *Principles of Mathematical Analysis*, McGraw–Hill, inc., 1976.

[SCH 86] Schrijver, A., *Theory of Linear and Integer Programming*, Wiley–Interscience, 1986.

[S 84] Spencer, J., "Guess a Number–with Lying", *Mathematics Magazine*, Vol. 57, 1984, pp. 105–108.

[S 65] Spivak, M., *Calculus on Manifolds*, Benjamin, New York, 1965.

[TS 81] Tenney, R.R. and Sandell, N.R. Jr., "Detection with Distributed Sensors", *IEEE Transaction on Aerospace and Electronic Systems*, AES–17, 4, 1981, pp. 501–510.

[T 79] Thompson, C.D., "Area Time Complexity for VLSI", *Proceedings of 11th STOC*, 1979, pp.81–88.

[Ti 84] Tiwari, P., "Lower Bounds on Communication Complexity in Distributed Computer Networks", *Proceedings of the 25th FOCS*, pp. 109–117, 1984.

[TW 80] Traub, J.F. and Wozniakowski, H., *A General Theory of Optimal Algorithms*, Academic Press, 1980.

[TWW 83] Traub, J.F., Wasilkowski, G.W. and Wozniakowski, H., *Information, Uncertainty, Complexity*, Addison–Wesley, 1983.

[Ts 85] Tsitsiklis, J.N., "Problems in Decentralized Decision Making and Computation", Ph.D. Dissertation, Technical Report, LIDS-TH-1424, Laboratory for Information and Decision Systems, M.I.T., 1985.

[TL 87] Tsitsiklis, J.N. and Luo, Z.Q., "Communication Complexity of Convex Optimization", *Journal of Complexity*, Vol. 3, pp. 231–243, 1987.

[U 76] Ulam, S.M., *Adventures of a Mathematician*, Scribner, New York, 1976.

[U 84] Ullman, J.D., *Computational Aspects of VLSI*, Computer Science Press, 1984.

[VB 81] Valiant, L.G. and Brebner, G.B., "Universal Schemes for Parallel Communication", *Proceedings of the 13th STOC,* 1981, pp. 263–277.

[VW 53] Van Der Waerden, B.L., *Modern Algebra* , Vol. 1 & 2, Frederick Ungar Publishing Co., New York, 1953.

[WB 82] Willsky, A.S., Bello, M.G., Castanon, D.A., Levy, B.C. and Verghese, G.C., "Combining and Updating of Local Estimates and Regional Maps Along Sets of One–Dimensional Tracks", *IEEE Transaction on Automatic Control*, Vol. AC-27, 4, 1982, pp. 799–812.

[Y 79] Yao, A.C., "Some Complexity Questions Related to Distributed Computing", *Proceedings of the 11th STOC,* 1979, pp. 209–213.

[Y 81] Yao, A.C., "The Entropic Limitations on VLSI Computations", *Proceedings of 13th STOC*, 1981, pp. 308–311.

[Z 83] Zak, F., "Projection of Algebraic Varieties", *Math. U.S.S.R. Sbornik*, Vol. 44, 1983, pp. 535–554.

[ZS 65] Zariski, O. and Samuel, P., *Commutative Algebra*, Vol. 1, D. Van Nostrand Company, inc., New Jersey, 1965.