Kathleen Steinhöfel (Ed.)

Stochastic Algorithms: Foundations and Applications

International Symposium, SAGA 2001
Berlin, Germany, December 13-14, 2001
Proceedings

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editor

Kathleen Steinhöfel
GMD - National Research Center for Information Technology
Institute for Computer Architecture and Software Engineering
Kekuléstr. 7, 12489 Berlin-Adlershof, Germany
E-mail: kathleen.steinhoefel@gmd.de

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Stochastic algorithms: foundations and applications : international symposium ; proceedings / SAGA 2001, Berlin, Germany, December 13 – 14, 2001 / Kathleen Steinhöfel (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 2264)
ISBN 3-540-43025-3

CR Subject Classification (1998):F.2, F.1.2, G.1.2, G.1.6, G.2, G.3

ISSN 0302-9743
ISBN 3-540-43025-3 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Steingräber Satztechnik GmbH, Heidelberg
Printed on acid-free paper SPIN 10846149 06/3142 5 4 3 2 1 0

Preface

SAGA 2001, the first Symposium on Stochastic Algorithms, Foundations and Applications, took place on December 13–14, 2001 in Berlin, Germany. The present volume comprises contributed papers and four invited talks that were included in the final program of the symposium.

Stochastic algorithms constitute a general approach to finding approximate solutions to a wide variety of problems. Although there is no formal proof that stochastic algorithms perform better than deterministic ones, there is evidence by empirical observations that stochastic algorithms produce for a broad range of applications near-optimal solutions in a reasonable run-time.

The symposium aims to provide a forum for presentation of original research in the design and analysis, experimental evaluation, and real-world application of stochastic algorithms. It focuses, in particular, on new algorithmic ideas involving stochastic decisions and exploiting probabilistic properties of the underlying problem domain. The program of the symposium reflects the effort to promote cooperation among practitioners and theoreticians and among algorithmic and complexity researchers of the field. In this context, we would like to express our special gratitude to DaimlerChrysler AG for supporting SAGA 2001.

The contributed papers included in the proceedings present results in the following areas: Network and distributed algorithms; local search methods for combinatorial optimization with application to constraint satisfaction problems, manufacturing systems, motor control unit calibration, and packing flexible objects; and computational learning theory.

The invited talk by Juraj Hromkovič surveys fundamental results about randomized communication complexity. In the talk, the efficiency of randomized communication is related to deterministic and nondeterministic communication models. Martin Sauerhoff discusses randomized variants of branching programs which allow the relative power of deterministic, nondeterministic, and randomized algorithms to be studied. Recent results on random 3-SAT formulas are summarized by Gregory Sorkin. The focus is on bounds for their density and shows how to tune so-called myopic algorithms optimally. Thomas Zeugmann gives an overview on stochastic finite learning that connects concepts from PAC learning and models of inductive inference learning.

Our special thanks go to all who supported SAGA 2001, to all authors who submitted papers, to all members of the program committee who provided very detailed referee reports, to the invited speakers, to the organizing committee, and to the sponsoring institutions.

December 2001 Kathleen Steinhöfel

Organization

SAGA 2001 was organized by the GMD - German Research Centre for Information Technology, Institute for Computer Architecture and Software Engineering FIRST, Berlin.

Organization Committee

Andreas Jakoby	Peggy Krüger
Babette Neumann	Friedrich Wilhelm Schröer
Kathleen Steinhöfel	Armin Wolf

Program Committee

Andreas Albrecht (University of Hertfordshire, UK)
Juraj Hromkovič (RWTH Aachen, Germany)
Oktay Kasim-Zade (Moscow State University, Russia)
Frieder Lohnert (DaimlerChrysler AG, Germany)
Evelyne Lutton (INRIA, France)
Dirk Mattfeld (University of Bremen, Germany)
Heinz Mühlenbein (GMD AIS, Germany)
Christian Scheideler (Johns Hopkins University, USA)
Gregory Sorkin (IBM Research, NY, USA)
Kathleen Steinhöfel (Chair, GMD FIRST, Germany)
Toby Walsh (University of York, UK)
Peter Widmayer (ETH Zurich, Switzerland)
CK Wong (The Chinese University, Hong Kong)
Thomas Zeugmann (Medical University of Lübeck, Germany)

Sponsoring Institutions

DaimlerChrysler AG, Germany
IT Service Omikron GmbH

Table of Contents

Randomized Communication Protocols (A Survey)

Juraj Hromkovič[*]

Lehrstuhl für Informatik I, RWTH Aachen,
Ahornstraße 55, 52074 Aachen, Germany

Abstract. There are very few computing models for which the power of randomized computing is as well understood as for communication protocols and their communication complexity. Since the communication complexity is strongly related to several complexity measures of distinct basic models of computation, there exist possibilities to transform some results about randomized communication protocols to other computing models, and so communication complexity has established itself as a powerful instrument for the study of randomization in complexity theory. The aim of this work is to survey the fundamental results about randomized communication complexity with the focus on the comparison of the efficiency of deterministic, nondeterministic and randomized communication.

Keywords: Randomized computing, communication complexity, two-party protocols.

1 Introduction

The communication complexity of two-party protocols was introduced by Yao [1] in 1978-79. (Note that communication complexity was implicitly considered by Abelson [2], too.) The initial goal was to develop a method for proving lower bounds on the complexity of distributed and parallel computations, with a special emphasis on VLSI computations [3–11].

But the study of communication complexity contributed to complexity theory much more than one had expected at the beginning in the early eighties and recently its relation to VLSI theory does not belong to the most important applications of communication protocols. The communication complexity theory established itself as a subarea of complexity theory and the main reasons for this successful story are the following ones:

(i) Communication complexity (similarly as Kolmogorov complexity) became a powerful method in proving lower bounds on fundamental complexity measures of concrete problems (see [3, 4] for some surveys). In some applications it caused a breakthrough in the long efforts in proving lower bounds (see,

[*] Supported by the DFG Project Hr.

for instance, [12–17]). This succeeded especially due to the development of a powerful nontrivial mathematical machinery for determining the communication complexity of concrete problems, while the relation of communication complexity to other complexity measures is typically transparent and usually easy to get.

(ii) Communication complexity essentially contributed to our understanding of randomized computation. There are very few computing models for which the power of randomized computing is as well understood as for two-party communication protocols. Moreover, due to (i) one can extend the results about randomized communication complexity to some other basic computing models.

This survey focuses on the study of randomized communication protocols. It is organized as follows. Section 2 presents the basic model of two-party communication protocols and its nondeterministic and randomized extensions. Here, we consider Las Vegas, one-sided-error and two-sided-error (bounded-error) randomization. Section 3 gives a short overview about methods for proving lower bounds on communication complexity. The central parts of this paper are Section 4 and 5. Section 4 relates the efficiency of randomized communication to the deterministic and nondeterministic communication models. Section 5 investigates whether one can restrict the number of random bits without restricting the power of randomized communication protocols. Because of the lack of space we do not present any survey of applications of results presented in Sections 4 and 5 to other computing models.

2 Definition of Communication Protocols

Informally, a **two-party (communication) protocol** consists of two computers C_I and C_{II} and a communication link between them. A protocol computes a finite function $f : U \times V \to Z$ in the following way. At the beginning C_I obtains an input $\alpha \in U$ and C_{II} obtains an input $\beta \in V$. Then C_I and C_{II} communicate according to the rules of the protocol by exchanging binary messages until one of them knows $f(\alpha, \beta)$. The **communication complexity of the computation on an input (α, β)** is the sum of the lengths of messages exchanged. The communication complexity of the protocol is the maximum of the complexities over all inputs from $U \times V$. The **communication complexity of f, cc(f)**, is the complexity of the best protocol for f.

Typically, one considers the situation that $U = \{0,1\}^n$, $V = \{0,1\}^n$ and $Z = \{0,1\}$, i.e. f is a Boolean function of $2n$ variables. Since this restriction is sufficient for our purposes we give the formal definition for protocols computing Boolean functions only. In what follows we describe the computation of a protocol on an specific input by a string $c_1\$c_2\$\ldots\$c_k\$c_{k+1}$, where $c_i \in \{0,1\}^+$ for $i = 1, \ldots, k$ are the messages exchanged (c_1 is the first message sent from C_I to C_{II}, c_2 is the second message submitted from C_{II} to C_I, etc.) and c_{k+1} is the result of the computation. The part $c_1\$c_2\$\ldots\$c_k$ is called the **history of communication**.

Definition 1. *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function over a set $X = \{x_1, x_2, \ldots, x_{2n}\}$ of Boolean variables. A* **protocol P over X** *(or* **over $\{0,1\}^n \times \{0,1\}^n$***) is a function from $\{0,1\}^n \times \{0,1,\$\}^*$ to $\{0,1\}^+ \cup \{\text{accept}, \text{reject}\}$ such that*

(i) P has the **prefix-freeness property***:*
For each $c \in \{0,1,\$\}^*$ and any two different $\alpha, \beta \in \{0,1\}^n$, $P(\alpha, c)$ is no proper prefix of $P(\beta, c)$.
{The next message $P(\alpha, c)$ is computed either by C_I or by C_{II} in the dependence of the input α of C_I (C_{II}) and the history c of communication. The prefix-freeness property assures that the messages exchanged between the two computers are self-delimiting, and no extra "end of transmission" symbol is required. To visualize the end of messages in the history of communication c, we write the special symbol \$ at the end of every message.}

(ii) If $\Phi(\alpha, c) \in \{\text{accept}, \text{reject}\}$ for an $\alpha \in \{0,1\}^m$, and $c \in (\{0,1\}^+\$)^{2p}$ for some $p \in \mathbb{N}$ [for $c \in (\{0,1\}^+\$)^{2p+1}$], then for all $q \in \mathbb{N}$, $\gamma \in \{0,1\}^m$, $d \in (\{0,1\}^+\$)^{2q+1}$ [$d \in (\{0,1\}^+\$)^{2q}$], $P(\gamma, d) \notin \{\text{accept}, \text{reject}\}$ for every communication history $d \in \{0,1,\$\}^*$.
{This property assures that the output value is always computed by the same computer independently of the input assignment.}

(iii) For every $c \in \{0,1,\$\}^*$, if $P(\alpha, c) \in \{\text{accept}, \text{reject}\}$ for some $\alpha \in \{0,1\}^n$, then $P(\beta, c) \in \{\text{accept}, \text{reject}\}$ for every $\beta \in \{0,1\}^n$.
{This property assures that if the computer C_I (C_{II}) computes the output for an input, then the other computer C_{II} (C_I) knows that C_I (C_{II}) knows the result, and so it does not wait for further communication.}

A **computation of P on an input (α, β)** $\in \{0,1\}^n \times \{0,1\}^n$ *is a string $c = c_1\$c_2\$\ldots\$c_k\$c_{k+1}$, where*

(1) $k \geq 0$, $c_1, \ldots, c_k \in \{0,1\}^+$, $c_{k+1} \in \{\text{accept}, \text{reject}\}$, and
(2) for every integer l, $0 \leq l \leq k$,
  (2.1) if l is even, then $c_{l+1} = P(\alpha, c_1\$c_2\$\ldots\$c_l\$)$, and
  {The message c_{l+1} is sent by C_I, and C_I computes c_{l+1} in dependence of its input part α and of the whole current communication history $c_1\$c_2\$\ldots\$c_l\$$.}
  (2.2) if l is odd, then $c_{l+1} = P(\beta, c_1\$c_2\$\ldots\$c_l\$)$.
  {The message c_{l+1} is sent from C_{II} to C_I, and C_{II} computes C_{l+1} in the dependence of its input β and the communication history $c_1\$c_2\$\ldots\$c_l\$$.}

For every computation $c = c_1\$c_2\$\ldots\$c_k\$c_{k+1}$,

$$\mathbf{Com(c)} = c_1\$c_2\$\ldots\$c_k\$$$

is the **communication** *(or* **communication history***) of c.*

We say that **P computes f** *if, for every $(\alpha, \beta) \in \{0,1\}^n \times \{0,1\}^n$, the computation of P on (α, β) is finite and ends with "*accept*" if and only if $f(\alpha, \beta) = 1$. In what follows we also say that a computation is* **accepting** *(***rejecting***) if it ends with* accept*(*reject*).*

The **length of a computation** c *is the total length of all messages in* c *(ignoring \$'s and the final* accept/reject*). The* **communication complexity of the protocol** $\boldsymbol{P}$, $\mathbf{cc}(\boldsymbol{P})$, *is the maximum of all computation lengths over all inputs* $(\alpha, \beta) \in \{0,1\}^n \times \{0,1\}^n$.

The **communication complexity of** $\boldsymbol{f}$ *is*

$$\mathbf{cc}(\boldsymbol{f}) = \min\{\mathrm{cc}(P) \mid P \textit{ computes } f\}.$$

□

We use the notation $\boldsymbol{P(\alpha, \beta)}$ for the output $\in$ {accept, reject} of the computation of the protocol P on the input (α, β). Note that "accept" is used to denote the result "1", and "reject" is used to denote the result "0". We use the notations "accept" and "reject" instead of "1" and "0" in order to distinguish between the communication bits and the results. Besides the reason above it will be convenient to be able to speak about accepting and rejecting computations in what follows.

To illustrate the above definition we consider the following simple examples. Let

$$\mathrm{Eq}_n(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n) = \bigwedge_{i=1}^{n} x_i \equiv y_i$$

be the Boolean function of $2n$ variables from $\{0,1\}^n \times \{0,1\}^n$ to $\{0,1\}$ that takes the value 1 iff the first half of the input is equal to the second half of the input. A protocol P computing Eq_n can simply work as follows. C_I sends its whole input $\alpha \in \{0,1\}^n$ to C_{II} and C_{II} compares whether α is identical with its input $\beta \in \{0,1\}^n$. Formally,

$P(\alpha, \lambda) = \alpha$, and[1]

$$P(\beta, \alpha\$) = \begin{cases} \text{accept} & \text{if } \alpha \equiv \beta. \\ \text{reject} & \text{if } \alpha \not\equiv \beta. \end{cases}$$

We observe that the above described strategy (C_I sends its whole input to C_{II}) works for every function and so

$$\mathrm{cc}(f) \leq n$$

for every Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ of $2n$ variables.

Now, consider the symmetric function $s_{2n} : \{0,1\}^{2n} \to \{0,1\}$ that takes the value 1 if and only if the number of 1's in the input is equal to the number of 0's in the input. A simple way to compute s_{2n} by a protocol follows. C_I sends the binary representation of the number $\#_1(\alpha)$ of 1's in its input $\alpha \in \{0,1\}^n$. C_{II} checks whether $\#_1(\alpha) + \#_1(\beta) = n$, where $\beta \in \{0,1\}^n$ is the input of C_{II}. Obviously, the communication complexity of this protocol is $\lceil \log_2(n+1) \rceil$.

Note that we shall later show that the above protocols for Eq_n and s_{2n} are optimal. We observe that these protocols are very simple because the whole

[1] λ denotes the empty word.

communication consists of the submission of one message. The protocols whose computations contains at most one message are called **one-way protocols** in what follows. For every Boolean function f

$$\mathbf{cc_1(f)} = \min\{\mathrm{cc}(P) \mid P \text{ is a one-way protocol computing } f\}$$

is the **one-way communication complexity of f**.

In Section 3 we shall see that there can be an exponential gap between $\mathrm{cc}_1(f)$ and $\mathrm{cc}(f)$. The following function $f_{\mathrm{ind}(n)}(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)$ for $n = 2^k$, $k \in I\!N - \{0\}$, is an example of a computing problem where one can profit from the exchange of more than one message between C_I and C_{II}. Let, for every binary string $\alpha = \alpha_0\alpha_1 \ldots \alpha_k$,

$$\mathbf{Number(\alpha)} = \sum_{i=0}^{k} \alpha_{k-i} \cdot 2^i$$

be the number with the binary representation α. The function $f_{\mathrm{ind}(n)}$ takes the value 1 iff

$$x_{\mathrm{Number}(y_{\mathrm{Number}(x_1 x_2 \ldots x_{\lceil \log_2 n \rceil})+1} \cdots y_{(\mathrm{Number}(x_1 x_2 \ldots x_{\lceil \log_2 n \rceil})+\lceil \log_2 n \rceil) \bmod n})+1} = 1.$$

Informally, the first $\log_2 n$ values of the variables $x_1, x_2, \ldots, x_{\log_2 n}$ determine a position (an index) $a = \mathrm{Number}(x_1, \ldots, x_{\log_2 n}) + 1$. y_a and the following $\log_2 n - 1$ values of y variables determine an index b, and one requires $x_b = 1$ for the result 1. We describe a protocol P that computes $f_{\mathrm{ind}(n)}$. C_I sends $x_1 x_2 \ldots x_{\lceil \log_2 n \rceil}$ to C_{II}. After that C_{II} sends the message $y_a y_{(a+1) \bmod n} \cdots y_{(a+\log_2 n - 1) \bmod n}$ to C_I, where $a = \mathrm{Number}(x_1 \ldots x_{\lceil \log_2 n \rceil}) + 1$. Now, C_I accepts iff

$$x_{\mathrm{Number}(y_a y_{(a+1) \bmod n} \cdots y_{(a+\lceil \log_2 n \rceil - 1) \bmod n})+1} = 1.$$

The communication complexity of this protocol is $2 \cdot \lceil \log_2 n \rceil$.

In what follows we use, for all nonnegative integers l, k, $l \geq \lceil \log_2 k \rceil$, $l \geq 1$, $\mathbf{BIN}_l\mathbf{(k)}$ to denote the binary representation of k by a binary string of length l. This means that if $l > \lceil \log_2 k \rceil$, $l - \lceil \log_2 k \rceil$ leading 0's are added to the representation.

One can introduce nondeterminism for protocols in the usual way. Because of this we prefer to give an informal description of nondeterministic protocols rather than an exact formal definition.

Let $f : U \times V \to \{0, 1\}$ be a finite function. A **nondeterministic protocol** P computing on inputs from $U \times V$ consists of two nondeterministic computers C_I and C_{II}. At the beginning C_I obtains an input $\alpha \in U$, and C_{II} obtains an input $\beta \in V$. As in the deterministic case, the computation consists of a number of communication rounds, where in one round one computer sends a message to the other one. The computation finishes when one of the computers decides to accept or to reject the input (α, β). In contrast to the deterministic case, C_I can be viewed as a relation on

$$(U \times \{0, 1, \$\}^*) \times (\{0, 1\}^+ \cup \{\text{accept,reject}\})$$

and C_{II} can be viewed as a relation on

$$(V \times \{0,1,\$\}^*) \times (\{0,1\}^+ \cup \{\text{accept,reject}\}).$$

This means that, for every argument (α, c), C_I (C_{II}) nondeterministically chooses a message from a finite set of possible messages determined by the argument (α, c).

We say that **P computes f** if for every $(\alpha, \beta) \in U \times V$,

(i) if $f(\alpha,\beta) = 1$, then there exists an accepting computation of P on the input (α,β), and
(ii) if $f(\alpha,\beta) = 0$, then all computations of P on (α,β) are rejecting ones.

Again, we require that the prefix-freeness property and the property that exactly one computer takes the final decision for all inputs are satisfied.

The **nondeterministic communication complexity of P**, denoted by **ncc(P)**, is the maximum of the lengths of all accepting computations of P. The **nondeterministic communication complexity of f** is

$$\mathbf{ncc}(f) = \min\{\text{ncc}(P) \mid P \text{ is a nondeterministic protocol computing } f\}.$$

To show the power of nondeterminism in communication consider, for every positive integer n, the function $\text{Ineq}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ defined by

$$\text{Ineq}_n(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n) = \bigvee_{i=1}^{n} (x_i \oplus y_i),$$

where $\oplus$ denotes the exclusive or (plus mod 2).

A nondeterministic protocol P that accepts all inputs $(\alpha,\beta) \in \{0,1\}^n \times \{0,1\}^n$ with $\alpha \neq \beta$ can work as follows. For every input $\alpha = \alpha_1\alpha_2\ldots\alpha_n$, C_I nondeterministically chooses a number $i \in \{1,\ldots,n\}$ and sends the message $\alpha_i\text{BIN}_{\lceil \log_2 n \rceil}(i)$ to C_{II}, where $\text{BIN}_{\lceil \log_2 n \rceil}(i)$ is the binary representation of i of the length[2] $\lceil \log_2 n \rceil$. Now, for every input $\beta = \beta_1 \ldots \beta_n$ of C_{II}, C_{II} accepts if and only if $\alpha_i \neq \beta_i$. In the case $\alpha_i = \beta_i$, C_{II} rejects the input. Obviously, for all $\alpha, \beta \in \{0,1\}^n$ with $\alpha \neq \beta$, there exists a j such that $\alpha_j \neq \beta_j$ and so there exists an accepting computation of P on (α,β). On the other hand if $\alpha = \beta$, then all n different computations of P on (α,β) are rejecting. So, P computes Ineq_n within the communication complexity $\lceil \log_2 n \rceil + 1$, i.e.,

$$\text{ncc}(\text{Ineq}_n) \leq \lceil \log_2 n \rceil + 1.$$

Similarly as in the deterministic case, one can consider **one-way nondeterministic protocols** whose computations contain at most one message. Let **ncc$_1$(f)** denote the **one-way nondeterministic communication complexity of f**. In contrast to the deterministic case, we show that there is no difference in the power of nondeterministic protocols and one-way nondeterministic protocols.

[2] This means, that additional 0's are taken to achieve the required length, if necessary.

Theorem 1. *For every finite function f,*

$$\mathrm{ncc}(f) = \mathrm{ncc}_1(f).$$

Proof. Let f be a function from $U \times V$ to $\{0,1\}$, and let $D = (C_I, C_{II})$ be a nondeterministic protocol computing f. We construct a one-way nondeterministic protocol D_1 that computes f within the communication complexity $\mathrm{ncc}(D)$. In what follows we say that a computation $C = c_1\$c_2\$\ldots\$c_k\$c_{k+1}$ is **consistent** for C_I with an input $\alpha \in U$ (or from the point of view of C_I and α) if C_I with the input α has the possibility to send the messages $c_1, c_3, c_5, \ldots$ when receiving the messages $c_2, c_4, c_6, \ldots$ from C_{II}. Similarly, one can define the consistency of C according to C_{II} with an input $\beta \in V$. Obviously, if C is a consistent computation for C_I on an input α and for C_{II} on an input β, then C is a possible computation of (C_I, C_{II}) on the input (α, β).

Let $C_1, C_2, \ldots, C_k$ be all accepting computations of D over all inputs from $U \times V$. Clearly, $k \leq 2^{\mathrm{ncc}(D)}$. Let D_1 consist of the computers C_I^1 and C_{II}^1. For every $\alpha \in U$, C_I^1 nondeterministically chooses an $i \in \{1, \ldots, k\}$, and it sends the message $\mathrm{BIN}_{\mathrm{ncc}(D)}(i)$ to C_{II}^1 if C_i is a possible accepting computation of D from the point of view of C_I and α. In other words, C_I^1 can send the binary representation of i if and only if there exists an $\gamma \in V$ such that C_i is the accepting computation of D on (α, γ). For every input β of C_{II}^1, when C_{II}^1 receives the binary representation of a number i, then C_{II}^1 accepts if C_i is a consistent accepting computation from the C_{II} and β point of view.

So, D_1 has exactly the same number of accepting computations as D and $\mathrm{ncc}_1(D_1) = \mathrm{ncc}(D)$. □

There are two distinct ways to introduce randomized protocols. One possibility is to take a nondeterministic protocol and to consider a probability distribution for every possible nondeterministic guess. Such randomized protocols are called **private** because each of the computers takes its random bits from a separate source; i.e., C_I (C_{II}) does not know the random bits of C_{II} (C_I). If one of the computers wants to know the random bits influencing the choice of the message submitted by the other computer, then these bits must be communicated. Another possibility to define randomized protocols is to say that a randomized protocol is a probability distribution over a set of deterministic protocols. Such randomized protocols are called **public** randomized protocols because this model corresponds to the situation when both computers have the same source of random bits (i.e. everybody sees the random bits of the other one for free). This second approach represents the well-known paradigm "*foiling an adversary*" of the design of randomized algorithms. So, every efficient public randomized protocol can be viewed as a successful application of this paradigm.

Clearly, public randomized protocols are at least as powerful as private ones. Newman [18] has proved that the relations between the communication complexities of public randomized protocols and private ones are linear for every bounded-error model. We shall present this result in Section 6. Because of this and in order to simplify the matters we formally define the public versions of randomized protocols only.

In the following definitions we use also the notation $P(\alpha, \beta) = 1$ (0) instead of $P(\alpha, \beta) =$ accept (reject).

Definition 2. *Let U and V be finite sets. A* **randomized protocol over $U \times V$** *is a pair $R = (Prob, S)$, where*

(i) $S = \{D_1, D_2, \dots, D_k\}$ is a set of (deterministic) protocols over $U \times V$, and
(ii) $Prob$ is a probability distribution over the elements of S.

For $i = 1, 2, \dots, k$, **$Prob(D_i)$** *is the probability that the protocol D_i is randomly chosen to work on a given input.*

For an input $(\alpha, \beta) \in U \times V$, the **probability that R computes an output z** *is*

$$\boldsymbol{Prob(R(\alpha,\beta) = z)} = \sum_{\substack{D \in \{D_1, \dots, D_k\} \\ D(\alpha,\beta) = z}} Prob(D).$$

The **communication complexity of R** *is*

$$\mathbf{cc}(\boldsymbol{R}) = \max\{\mathrm{cc}(D) \mid D \in S\}.$$

A randomized protocol $R = (Prob, S)$ is called **one-way** *if all elements of S are one-way protocols.*

For every randomized protocol $R = (Prob, \{D_1, D_2, \dots, D_k\})$ with a uniform probability distribution $Prob$, the **degree of randomness** *of P is $\lceil \log_2 k \rceil$. Since one can unambiguously identify every D_i with a binary string of length $\lceil \log_2 k \rceil$, we call $\lceil \log_2 k \rceil$ the* **number of random bits of R**, *too.*

In what follows we consider only randomized protocols computing Boolean functions. In contrast to the previous protocol models we also allow a "**neutral**" output "?", whose meaning is that the protocol was not able to compute any final answer in the given computation (random attempt). We consider this possibility for Las Vegas protocols that never err but may produce the answer "?" with a bounded probability.

Note that in what follows we use also the notation $R(\alpha, \beta) = 1$ ($R(\alpha, \beta) = 0$) instead of $R(\alpha, \beta) =$ "accept" ($R(\alpha, \beta) =$ "reject"). This is convenient because we obtain the possibility to speak about the probability of the event $R(\alpha, \beta) = f(\alpha, \beta)$ in this way.

Definition 3. *Let $f : U \times V \to \{0, 1\}$ be a finite function. We say, that a randomized protocol $R = (Prob, S)$ is a* **(public) Las Vegas protocol for f** *if*

(i) for every $(\alpha, \beta) \in U \times V$ with $f(\alpha, \beta) = 1$,
$Prob(R(\alpha, \beta) = 1) \geq \frac{1}{2}$ and $Prob(R(\alpha, \beta) = 0) = 0$, and
(ii) for every $(\alpha, \beta) \in U \times V$ with $f(\alpha, \beta) = 0$,
$Prob(R(\alpha, \beta) = 0) \geq \frac{1}{2}$ and $Prob(R(\alpha, \beta) = 1) = 0$.

The **Las Vegas communication complexity of f** *is*

$$\mathbf{lvcc}(\boldsymbol{f}) = \min\{\mathrm{cc}(R) \mid R \textit{ is a Las Vegas protocol for } f\}.$$

The **one-way Las Vegas communication complexity of f** *is*

$$\mathbf{lvcc_1}(\boldsymbol{f}) = \min\{\mathrm{cc}(R) \mid R \textit{ is a one-way Las Vegas protocol for } f\}.$$

We present a simple example of a one-way Las Vegas protocol. More involved ideas for the design of Las Vegas protocols can be found in Section 4. Consider the function $\mathrm{Index}_n : \{0,1\}^n \times \{1,2,\ldots,n\} \to \{0,1\}$ defined as follows[3]

$$\mathrm{Index}_n((x_1,x_2,\ldots,x_n),j) = x_j.$$

A Las Vegas one-way protocol D for Index_n can be described as the pair $(Prob, \{D_1, D_2\})$, where

(i) $Prob(D_1) = Prob(D_2) = \frac{1}{2}$,
(ii) $D_1 = (D_{I,1}, D_{II,1})$, where $D_{I,1}$ sends the first half $\alpha_1 \ldots \alpha_{\lceil \frac{n}{2} \rceil}$ of its input $\alpha_1 \ldots \alpha_n$ to $D_{II,1}$. $D_{II,1}$ outputs α_j if the input j of $D_{II,1}$ belongs to $\{1,2,\ldots,\lceil \frac{n}{2} \rceil\}$. If $j > \lceil \frac{n}{2} \rceil$, then $D_{II,1}$ outputs "?",
(iii) $D_2 = (D_{I,2}, D_{II,2})$ where $D_{I,2}$ sends the second half $\alpha_{\lceil \frac{n}{2} \rceil+1} \ldots \alpha_n$ of its input $\alpha_1 \ldots \alpha_n$ to $D_{II,2}$. $D_{II,2}$ outputs α_j if the input j of $D_{II,2}$ belongs to $\{\lceil \frac{n}{2} \rceil + 1, \ldots, n\}$. If $j \le \lceil \frac{n}{2} \rceil$, then $D_{II,2}$ outputs "?".

Another possibility to describe D is as follows.

Las Vegas one-way protocol $D = (D_I, D_{II})$ for Index_n.

```
Input: $(\alpha, j)$, $\alpha = \alpha_1 \ldots \alpha_n \in \{0,1\}^n$, $j \in \{1,\ldots,n\}$.
    { $D_I$ gets the input $\alpha$, and $D_{II}$ gets the input $j$.}
Step 1: $D_I$ chooses a random bit $r \in \{0,1\}$.
    { Note, that $D_{II}$ knows $r$, too.}
    If $r = 0$, then $D_I$ sends the message $\alpha_1\alpha_2 \ldots \alpha_{\lceil \frac{n}{2} \rceil}$.
    If $r = 1$, then $D_I$ sends the message $\alpha_{\lceil \frac{n}{2} \rceil+1}\alpha_{\lceil \frac{n}{2} \rceil+2} \ldots \alpha_n$.
Step 2: If $r = 0$ and $j \in \{1,2,\ldots,\lceil \frac{n}{2} \rceil\}$, then $D_{II}$ outputs $\alpha_j$.
    If $r = 1$ and $j > \lceil \frac{n}{2} \rceil$, then $D_{II}$ outputs $\alpha_j$.
    Else, $D_{II}$ outputs "?".
```

In what follows we shall prefer the second form of the description of randomized protocols. Clearly, D never errs, and the probability of giving the output "?" is $\frac{1}{2}$ for every input $(\alpha, j) \in \{0,1\}^n \times \{1,\ldots,n\}$. The communication complexity of D is $\lceil \frac{n}{2} \rceil$.

Note, that the constant $\frac{1}{2}$ bounding the probability of the output "?" in the definition of (two-way) Las Vegas protocols is not essential from the asymptotic point of view. Instead of giving the output "?" a Las Vegas protocol may start a new communication from the beginning with new random bits. If it outputs "?" only if it reaches "?" in k independent computation attempts, then the probability to obtain the output "?" decreases from $\frac{1}{2}$ to $\frac{1}{2^k}$, but the communication complexity increases only by a factor of k in comparison with the original protocol.

Definition 4. *Let $f : U \times V \to \{0,1\}$ be a finite function. We say that a randomized protocol $R = (Prob, S)$ is a* **(public) one-sided-error Monte Carlo protocol for f** *if*

[3] Observe, that Index_n can be also viewed as a Boolean function if one represents the numbers $1, 2, \ldots, n$ by binary strings.

(i) for every $(\alpha, \beta) \in U \times V$ *with* $f(\alpha, \beta) = 1$,
Prob$(R(\alpha, \beta) = 1) \geq \frac{1}{2}$, *and*
(ii) for every $(\alpha, \beta) \in U \times V$ *with* $f(\alpha, \beta) = 0$,
Prob$(R(\alpha, \beta) = 0) = 1$.

We say that a randomized protocol R is a **(public) two-sided-error Monte Carlo protocol for *f*** *if, for every* $(\alpha, \beta) \in U \times V$,

$$Prob(R(\alpha, \beta) = f(\alpha, \beta)) > \frac{2}{3}.$$

The **one-sided-error Monte Carlo communication complexity of *f*** *is*

$$\mathbf{1mccc}(f) = \min\{\mathrm{cc}(R) \mid R \textit{ is a one-sided-error Monte Carlo protocol for } f\}.$$

The **two-sided-error Monte Carlo communication complexity of *f*** *is*

$$\mathbf{2mccc}(f) = \min\{\mathrm{cc}(R) \mid R \textit{ is a two-sided-error Monte Carlo protocol for } f\}.$$

Because of the condition (ii) of one-sided-error Monte Carlo protocols it is clear that private one-sided-error Monte Carlo protocols are a restricted version of nondeterministic ones. For the public randomized protocols defined here we obtain

$$\mathrm{ncc}(f) \leq 1\mathrm{mccc}(f) + \text{ the number of random bits}$$

for every finite function f.

Similarly as in the case of Las Vegas, the constant $\frac{1}{2}$ in the inequality $Prob(R(\alpha, \beta) = 1) \geq \frac{1}{2}$ is not essential for one-sided-error Monte Carlo from the asymptotic point of view and so one-sided-error Monte Carlo protocols can be viewed as a restricted version of two-sided-error Monte Carlo protocols, too.

Example 1. (based on [19]) The idea of a randomized protocol is based on the **abundance of witnesses** method for the design of randomized algorithms. Let f be a Boolean function. A **witness for $f(\gamma) = a$** is any binary string δ, such that using δ there is an efficient way to prove (verify) that $f(\gamma) = a$. For instance, any factor (nontrivial divisor) y of a number x is a witness of the claim "*x is composite*". Obviously, to check whether $x \bmod y = 0$ is much easier than to prove that x is a composite without any additional information. In general, one considers witnesses only if they essentially decrease the complexity of computing the result. For many functions the difficulty with finding a witness deterministically is that the witness lies in a search space that is too large to be searched exhaustively. However, by establishing that the space contains a large number of witnesses, it often suffices to choose an element at random from the space. The randomly chosen item is likely to be a witness. If this probability is not high enough, an independent random choice of several items reduces the probability that no witness is found.

The framework of this approach is very simple. One has for every input γ a set CandW(γ) that contains all items candidating to be a witness for the input γ. Often CandW(γ) is the same set for all inputs of the same size as γ. Let Witness(γ) contain all witnesses for γ that are in CandW(γ). The aim is to reach a situation where the cardinality of Witness(γ) is proportional to the cardinality of CandW(γ).

To design a randomized protocol for Ineq_n, we say that a **prime p is a witness for $\alpha \neq \beta$**, $\alpha, \beta \in \{0,1\}^n$, if

$$\mathrm{Number}(\alpha) \bmod p \neq \mathrm{Number}(\beta) \bmod p.$$

For every input $(\alpha, \beta) \in \{0,1\}^n \times \{0,1\}^n$, CandW($\alpha, \beta$) is the set of all primes from $\{2, 3, \ldots, n^2\}$. Due to the Prime Number Theorem we know that $|\mathrm{CandW}(\alpha, \beta)|$ is approximately $\frac{n^2}{\ln n^2}$. Now, we estimate the lower bound on $|\mathrm{Witness}(\alpha, \beta)|$. Let $\alpha \neq \beta$. If, for a prime p,

$$\mathrm{Number}(\alpha) \bmod p = \mathrm{Number}(\beta) \bmod p, \tag{1}$$

then p divides $h = \mathrm{Number}(\alpha) - \mathrm{Number}(\beta)$. Since $h < 2^n$, h has fewer than n different prime divisors.[4] This means that at most $n-1$ primes from CandW(α, β) have the property (1). So,

$$|\mathrm{Witness}(\alpha, \gamma)| \geq |\mathrm{CandW}(\alpha, \beta)| - n + 1. \tag{2}$$

Now, we use (2) to design our randomized protocol.

One-sided-error Monte Carlo protocol $R = (R_I, R_{II})$ for Ineq_n.

Input: $(\alpha, \beta) \in \{0,1\}^n \times \{0,1\}^n$
Step 1: R_I chooses uniformly a prime $p \in \{2, 3, \ldots, n^2\}$ at random.
{Note, that R_{II} knows this choice of R_I.}
Step 2: R_I computes $s = \mathrm{Number}(\alpha) \bmod p$ and sends the binary representation of s to R_{II}.
{Note, that the length of the message is $\lceil \log_2 n^2 \rceil \leq 2 \cdot \lceil \log_2 n \rceil$.}
Step 3: R_{II} computes $q = \mathrm{Number}(\beta) \bmod p$.
If $q \neq s$, then R_{II} outputs 1 ("accept").
If $q = s$, then R_{II} outputs 0 ("reject").

We show that R is a one-sided-error Monte Carlo protocol for Ineq_n. If $\alpha = \beta$, for an input $(\alpha, \beta) \in \{0,1\}^n \times \{0,1\}^n$, then $\mathrm{Number}(\alpha) \bmod p = \mathrm{Number}(\beta) \bmod p$ for every prime p. So,

$$Prob(R(\alpha, \beta) = \text{"reject"}) = 1.$$

Let $\alpha \neq \beta$, i.e. $\mathrm{Ineq}_n(\alpha, \beta) = 1$. Due to the inequality (2), the probability that R chooses a prime with the property (1) is at most

$$\frac{|\mathrm{CandW}(\alpha, \beta)| - |\mathrm{Witness}(\alpha, \gamma)|}{|\mathrm{CandW}(\alpha, \beta)|} \leq \frac{n-1}{|\mathrm{CandW}(\alpha, \beta)|}.$$

[4] Observe, that $n! > 2^n$.