Electronic Colloquium on Computational Complexity, Report No. 62 (2004)

# A note on the P versus NP intersected with co-NP question in communication complexity

Stasys Jukna *†‡

**Abstract**

We consider the P versus NP ∩ co-NP question for the classical two-party communication protocols: if both a boolean function $f$ and its negation $\neg f$ have small nondeterministic communication complexity, what is then its deterministic and/or probabilistic communication complexity? In the fixed (worst) partition case this question was answered by Aho, Ullman and Yannakakis in 1983: here P = NP ∩ co-NP.

We show that in the best-partition case the situation is entirely different: here P is a proper subset of NP ∩ co-NP. This resolves an open question raised by Papadimitriou and Sipser in 1982. Actually, we prove even stronger separations: P is a proper subset of RP ∩ co-RP, and NP ∩ co-NP is no longer a subset of BPP.

## 1 Introduction

Understanding the relative power of determinism, nondeterminism, and randomization is fundamental in any model of computation. In the Turing machine model this leads to the well-known P versus NP versus BPP and similar questions. While in this model such questions remain widely open, some progress was made in simpler (but still fundamental) models, like decision trees or communication protocols.

In the decision tree model when the complexity measure is the *depth* of a tree we have that P = NP ∩ co-NP [3, 7, 19] and even P = BPP [15]. Interestingly, the situation is different if we measure the *size* of a tree instead of its depth: then P ≠ NP ∩ co-NP [11]. Similar results were obtained for various restricted models of branching programs—a generalization of decision trees. In particular, for read-once branching programs we have P ≠ NP ∩ co-NP [11] and even NP ∩ co-NP ⊄ BPP [18].

In this paper we consider the classical model of two-party *best-partition* communication protocols and show that here P ≠ NP ∩ co-NP. Actually, we establish stronger separations

*Universität Frankfurt, Institut für Informatik, Robert-Mayer-Str. 11-15, D-60054 Frankfurt am Main, Germany & Institute of Mathematics and Informatics, Akademijos 4, LT-2600 Vilnius, Lithuania

†*Email*: jukna@thi.informatik.uni-frankfurt.de

‡Research supported in part by a DFG grant SCHN 503/2-1.



ISSN 1433-8092

in this model: here $\mathrm{P} \subsetneqq \mathrm{RP} \cap \text{co-RP} \subsetneqq \mathrm{NP} \cap \text{co-NP} \not\subseteq \mathrm{BPP}$. This, in particular, resolves an open question of Papadimitriou and Sipser [16] and contrast with the situation in the *fixed-partition* case where $\mathrm{P} = \mathrm{NP} \cap \text{co-NP} \subsetneqq \mathrm{BPP}$ [1, 21]. Proofs themselves are amazingly simple—as it often happens with the results of this type, most of the work is usually done by a careful choice of separating functions.

# 2 The model

The model we consider is the classical model of two-party communication protocols introduced by Yao in [20] (see also the monographs [8, 13] for more information). Recall that in this model, the set of variables $X$ is partitioned in two parts whose sizes differ by at most one; such a partition is called *balanced*. There are two cooperative but distant players, Alice and Bob, who wish to compute a given boolean function $f(X)$. Each player has unlimited computational power and full knowledge of the function. However, each player has only partial information about the input: Alice has access only to the variables in the first part and Bob only to the variables in the second part of the partition of $X$. The objective is to compute the function with as small an amount of communication as possible. The measure of complexity is the amount of communication on the worst-case input.

After the partition of the set of variables $X$ is fixed, the players must use it for all inputs. The subtle point, however, is whether the players can choose the (most convenient for a given function) partition—this is the *best-partition* (called also *variable-partition* in [13]) case—or they must use some fixed in advanced "bad" partition—this is the *fixed-partition* case. The best way to see the difference is to consider the equality function $\mathrm{EQ}(x, y)$ which tests whether two given binary strings $x$ and $y$ of length $n$ are equal. If Alice gets $x$ and Bob $y$ then the best what the players can do is that Alice sends her entire string $x$ to Bob—it can be easily shown that no protocol can do better. On the other hand, if Alice takes the first $n/2$ bits of $x$ and $y$, and Bob takes the rest, then 2 bits of communication are already enough.

There are also three natural modes of communication: deterministic, nondeterministic and probabilistic. In a *nondeterministic* communication game the protocol first guesses some binary string, tells this string to both players, and players then communicate deterministically according to that string. In a *probabilistic* protocol Alice and Bob are allowed to (privately) flip a random coin. That is, in this case for each input the messages may depend on two strings chosen independently, according to some probability distribution. Such a protocol computes a given function with *bounded error* if for every input it outputs a correct value with probability at least 2/3. That is, such a protocol may err on both accepted and rejected inputs. A probabilistic protocol computes $f$ with *one-sided error* if it rejects each input in $f^{-1}(0)$ with probability 1 and accepts each input in $f^{-1}(1)$ with probability at least 2/3.

We adopt the following convention for discussing different communication complexity measures of $f$ in the *best-partition* case: $D(f)$ for the deterministic complexity, $N(f)$ for

the nondeterministic complexity, $R(f)$ for the probabilistic bounded error complexity, and $R^1(f)$ for probabilistic one-sided error communication complexity.

# 3 Previous work

Having three modes—deterministic, nondeterministic and probabilistic—and having the (admittedly far-fetched) analogy with the P versus NP question, one may ask whether, say, NP = co-NP or P = NP ∩ co-NP or P = RP ∩ co-RP or NP ∩ co-NP ⊆ BPP in the context of communication protocols. Here we use the common names for the analogs of the complexity classes: P (resp., NP, BPP and RP) consists of all boolean functions in $n$ variables whose deterministic (resp., nondeterministic, probabilistic bounded error, and probabilistic one-sided error) communication complexity is polynomial in $\log n$. It is clear that P ⊆ RP ⊆ NP∩BPP in both the fixed- and the best-partition case.

In the *fixed-partition* case most of these problems are already solved. In particular, the following separations are known. (Of course, this has nothing to do with the relations between Turing machine classes.)

- NP ≠ co-NP. This can be easily shown using the equality function $\textsc{eq}(x, y)$.
- BPP ⊈ NP and P ≠ RP. This was proved by Yao [21]: $\neg\textsc{eq}(x, y)$ can be computed by a probabilistic (even one-sided $1/n$-error) protocol with only $O(\log n)$ bits of communication.
- NP ⊈ BPP. This was proved by Babai, Frankl, and Simon [2] using the set-disjointness function $\textsc{disj}(x, y)$, which outputs 1 iff $\sum_{i=1}^{n} x_i y_i = 0$. They proved that this function has probabilistic communication complexity $\Omega(\sqrt{n})$. This lower bound was improved to $\Omega(n)$ by Kalyanasundaram and Schnitger [12]; a simpler proof was found by Razborov [17].

Interestingly, the P versus NP ∩ co-NP question in the fixed-partition model has a positive answer:

$$\mathrm{P} = \mathrm{RP} \cap \text{co-RP} = \mathrm{NP} \cap \text{co-NP} \subsetneq \mathrm{BPP}.$$

This is a direct consequence from the following (somewhat surprising) result of Aho, Ullman, and Yannakakis [1].

**Theorem 1 ([1]).** *If $f$ and $\neg f$ have nondeterministic communication complexities $n_f$ and $n_{\neg f}$, then the deterministic communication complexity of $f$ does not exceed $O(\max\{n_f, n_{\neg f}\}^2)$.*

This result was later improved in different ways. Halstenberg and Reischuk [6] have strengthened the upper bound to the form $O(n_f \cdot n_{\neg f})$. Lovász and Saks [14] have strengthened this last upper bound by showing that $n_f$ can be replaced with the triangular rank of the corresponding communication matrix of $f$. The tightness of the theorem was proved by

Fürer [5] using so-called "list inequality" function. This was recently improved by Jayram, Kumar and Sivakumar [9] by showing that, for the iterated set-disjointness function $f$, even the probabilistic bounded error communication complexity is $\Omega(n_f \cdot n_{\neg f})$.

The *best-partition* model is more difficult to analyze, and here the situation was less clear. The first non-trivial separation in this case was proved by Papadimitriou and Sipser [16]. Using the triangle-freeness property of graphs and an elegant combinatorial argument, they proved that NP $\neq$ co-NP also in this case. The proof was via a reduction to computing $\neg\textsc{disj}(x,y)$ in the fixed-partition case. Using probabilistic arguments, this result was extended in [10] to the case where a protocol is allowed to use different partitions for different inputs (the logarithm of the number of used partitions is, however, a part of the complexity): even in this model the triangle-freeness function has exponentially high nondeterministic communication complexity. This, together with the fact, proved in [4], that using $k+1$ partitions instead of $k$ partitions may exponentially decrease the number of communicated bits, shows that in the context of communication complexity the corresponding classes NP and co-NP are indeed very different.

In the same paper [16] (this was one of the first papers to study questions of this type in communication complexity) Papadimitriou and Sipser asked whether P $\neq$ NP$\cap$co-NP for the best-partition protocols. The question is important because it exposes something about the power of lower bound arguments. That is, fooling set and other arguments, used for the best-partition protocols, apply not only to deterministic but also to nondeterministic protocols. Since $D(\neg f) = D(f)$, we can prove a lower bound on $D(f)$ by arguing about either $f$ or $\neg f$. But if *both* the function and its negation have low nondeterministic complexity under some partitions of variables, other arguments are needed to show that $D(f)$ must be large for *any* partition.

To our best knowledge this question remained unsolved or, at least, its solution was not published. The only result in this direction we are aware of is the claim in [1] that an appropriate modification of the triangle-freeness function should separate P from NP $\cap$ co-NP in the best-partition case. But the proof—which should (apparently) involve the argument for the triangle-freeness function used in [16]—was not given.

# 4 The result

In this paper we prove that P $\neq$ NP $\cap$ co-NP for the best-partition protocols. Actually, we establish even stronger separations P $\neq$ RP $\cap$ co-RP and NP $\cap$ co-NP $\not\subseteq$ BPP. Thus, in the best-partition case the situation is entirely different: here we have that

$$\mathrm{P} \subsetneq \mathrm{RP} \cap \text{co-RP} \subsetneq \mathrm{NP} \cap \text{co-NP} \not\subseteq \mathrm{BPP}.$$

This is a direct consequence of the following theorem.

**Theorem 2.** *There are explicit boolean functions $f$ and $g$ in $n^2$ variables such that:*

(i) *both $N(f)$ and $N(\neg f)$ are constants but $R(f) = \Omega(n)$;*

(ii) *both* $R^1(g)$ *and* $R^1(\neg g)$ *are* $O(\log n)$ *but* $D(g) = \Omega(n)$.

# 5 Proof of Theorem 2(i)

We define the boolean function separating NP $\cap$ co-NP from BPP in the best-partition case as follows. The function $f(X)$ (let us call it the *good matrix function*) has $n^2$ variables where $n$ is a sufficiently large positive integer. Assume that the set of input variables $X$ is arranged into an $n \times n$ matrix. Hence, inputs for $f$ are 0/1 matrices $A : X \to \{0,1\}$. Say that a row/column $x$ of such a matrix is *good* if it contains precisely two 1's, and *bad* otherwise. Let $f(A) = 1$ if and only if

(i) at least one row of $A$ is good, and

(ii) all columns of $A$ are bad.

**Lemma 1.** *Both* $N(f)$ *and* $N(\neg f)$ *are constants.*

*Proof.* To compute $f(X)$ the players take a partition of $X$ where Alice gets the first half of the *columns* and Bob gets the rest. Given an input matrix $A : X \to \{0,1\}$, the protocol first guesses a row $r$ (a candidate for a good row). Then, using 3 bits, Alice tells Bob whether all her columns are bad, and whether the first half of the row $r$ contains none, one, two or more 1's. After that Bob has the whole information about the value $f(A)$, and can announce the answer.

In order to compute $\neg f(X)$ the players take a partition of $X$ where Alice gets the first half of the *rows* and Bob gets the rest. Given an input matrix $A : X \to \{0,1\}$, the protocol first guesses a column $c$ (a candidate for a good column). Then, using 3 bits, Alice tells Bob whether there is a good row among her rows, and whether the first half of the column $c$ contains none, one, two or more 1's. After that Bob again has the whole information about the value $f(A)$, and can announce the answer. □

In the proof of a lower bound on $R(f)$ we will use the fact (mentioned in Sect. 3) that the set-disjointness function DISJ$(x,y)$, which outputs 1 iff $\sum_{i=1}^{n} x_i y_i = 0$, has high fixed-partition communication complexity even in probabilistic protocols: if Alice gets $x$ and Bob gets $y$ then $R(\text{DISJ})$ is $\Omega(\sqrt{n})$ [2], and even $\Omega(n)$ [12, 17]. We will also use the (obvious) fact that $R(f) = R(\neg f)$ for every boolean function $f$.

**Lemma 2.** $R(f) = \Omega(n)$.

*Proof.* Take an arbitrary probabilistic bounded error protocol for $f(X)$. The protocol uses some balanced partition of $X$ into two halves where the first half is seen by Alice and the second by Bob. Say that a column is seen by Alice (resp., Bob) if Alice (resp., Bob) can see all its entries. A column is *mixed* if it is seen by none of the two players, that is, if each player can see at least one its entry. Let $m$ be the number of mixed columns. We consider two cases depending on how large this number $m$ is. In both cases we describe a "hard"

subset of inputs, i.e. a subset of input matrices on which the players need to communicate many bits.

*Case 1:* $m \leq n/2 - 1$. Since each player can see at most $n/2$ columns, we have that in this case each player will see at least $n - (n/2 + m) \geq 1$ columns. Take one column seen by Alice and another column seen by Bob, and let $Y$ be the $(n-3) \times 2$ submatrix of $X$ formed by these two columns without the last three rows. We restrict the protocol to input matrices $A : X \to \{0,1\}$ defined as follows. We first set all entries in the last three rows to 1. This way we ensure that all columns of $A$ are already bad. Then we set all remaining entries of $X$ outside $Y$ to 0. The columns $x$ and $y$ of $Y$ may take arbitrary values.

In each such matrix all columns are bad and, since $n \geq 3$, the last three all-1 rows are also bad. Thus, given such a matrix, the players must determine whether some of the remaining rows is good. Since all these rows have 0's outside the columns $x$ and $y$, this means that the players must determine whether $x_i = y_i = 1$ for some $1 \leq i < n - 3$. That is, they must compute $\neg\textsc{disj}(x, y)$ which requires $\Omega(n)$ bits of communication.

*Case 2:* $m \geq n/2$. Let $Y$ be the $n \times m$ submatrix of $Y$ formed by the mixed columns. Select from the $i$-th $(i = 1, \ldots, m)$ column of $Y$ one entry $x_i$ seen by Alice and one entry $y_i$ seen by Bob. Since $m \leq n$ and we select only $2m$ entries, there must be a row $r$ with $t \leq 2$ selected entries. Let $Y$ be the $n \times (m-t)$ submatrix consisting of the mixed columns with no selected entries in the row $r$. We may assume that $m - t$ is odd and that $m - t \leq n - 2$ (if not, then just include in $Y$ fewer columns).

Now restrict the protocol to input matrices $A : X \to \{0,1\}$ defined as follows. First we set to 1 some two entries of the row $r$ lying outside $Y$, and set to 0 all the remaining entries of $r$. This ensures that the obtained matrices will already contain a good row. After that we set all the remaining non-selected entries of $X$ to 0. Since each obtained matrix $A$ contains a good row (such is the row $r$) and all columns outside the submatrix $Y$ are bad (each of them can have a 1 only in the row $r$), the players must determine whether all columns of $A$ in $Y$ are also bad. Since all non-selected entries of $Y$ are set to 0, this means that the players must determine whether $x_i + y_i \leq 1$ for all $i = 1, \ldots, m - t$. That is, they must decide whether $\sum_{i=1}^{m-t} x_i y_i = 0$, i.e. to compute the set-disjointness function $\textsc{disj}(x, y)$, which again requires $\Omega(m - t) = \Omega(n)$ bits of communication.

This completes the proof of Lemma 2, and thus, the proof of Theorem 2(i). □

# 6 Proof of Theorem 2(ii)

The desired separating function $g(X)$ (let us call it the *odd-even function*) is defined in a similar way as the good matrix function used in the proof above. As before, inputs for $g(X)$ are $n \times n$ matrices; this time we require that $n$ is even. Say that a row/column of such a matrix is *odd* (*even*) if it contains an odd (even) number of 1's. Let $g(A) = 1$ if and only if

(i) $A$ has at least one odd row, and

(ii) all columns of $A$ are odd.

The *nonequality function* is a boolean function $\text{NE}(x,y)$ in $2n$ variables such that $\text{NE}(x,y) = 1$ iff $x \neq y$, i.e. is the strings $x$ and $y$ differ in at least one coordinate. It is known (see [21] or Example 3.9 in [13]) that $R^1(\text{NE}) = O(\log n)$.

**Lemma 3.** *Both $R^1(g)$ and $R^1(\neg g)$ are $O(\log n)$.*

*Proof.* Using the same partitions of $X$ as in the proof of Lemma 1, we see that the computation of the odd-even function $g$ and its negation $\neg g$ reduces to the computation of the nonequality function $\text{NE}(x,y)$, where $x$ is a string of parities of rows/columns seen by Alice and $y$ is a string of parities of rows/columns seen by Bob. Indeed, to compute $g$ it is enough to decide whether $x \neq y$ (there is an odd row) whereas for $\neg g$ it is enough to decide whether $x \neq y \oplus \mathbf{1}$ (there is an even column). □

**Lemma 4.** $D(g) = \Omega(n)$.

*Proof.* The idea is the same as in the proof of Lemma 2. Still there are some technical differences, so let us give the whole proof. Take an arbitrary deterministic protocol for $g(X)$ using some balanced partition of $X$. As before, let $m$ be the number of mixed columns.

*Case 1:* $m \leq n/2-1$. In this case each player can see at least one column. Take one column seen by Alice and another column seen by Bob, and let $Y$ be the $(n-1) \times 2$ submatrix of $X$ formed by these two columns without the last row $r$. We restrict the protocol to input matrices $A : X \to \{0,1\}$ defined as follows. We set to 1 all entries in the last row $r$, and set to 0 all remaining entries of $X$ outside $Y$. The columns $x$ and $y$ of $Y$ may take arbitrary values such that the resulting vectors are even. This way we ensure that all columns of $A$ are odd. Moreover, the last row $r$ is even since $n$ is even. Thus, given such a matrix $A$, the players must determine whether some of the remaining rows is odd. That is, they must determine whether $x \neq y$, which requires $\Omega(n)$ bits of communication.

*Case 2:* $m \geq n/2$. Let $Y$ be the $n \times m$ submatrix of $Y$ formed by the mixed columns. Select from the $i$-th $(i = 1, \ldots, m)$ column of $Y$ one entry $x_i$ seen by Alice and one entry $y_i$ seen by Bob. Since $m \leq n$ and we select only $2m$ entries, there must be a row $r$ with $t \leq 2$ selected entries. Let $Y$ be the $n \times (m-t)$ submatrix $(t \leq 2)$ consisting of the mixed columns with no selected entries in this row $r$. We may assume that $m-t$ is odd (if not, then just include one column less in $Y$).

Now restrict the protocol to input matrices $A : X \to \{0,1\}$ defined as follows. First we set the part of the row $r$ lying in $Y$ to 0's and the rest of $r$ to 1's. Since $n$ is even and $m-t$ is odd, this ensures that the obtained matrices will already contain an odd row. After that we set to 0 all the remaining non-selected entries of $X$. Since each obtained matrix $A$ contains an odd row (the row $r$) and all columns outside the submatrix $Y$ are odd (each of them has a 1 in the row $r$ and 0's elsewhere), the players must determine whether all columns of $A$ in $Y$ are also odd. That is, they must determine whether $x_i \neq y_i$ for all $i = 1, \ldots, m-t$.

Or equivalently, they must decide whether $x = y \oplus \mathbf{1}$ for vectors $x = (x_1, \ldots, x_{m-t})$ and $y = (y_1, \ldots, y_{m-t})$, which again requires $\Omega(m-t) = \Omega(n)$ bits of communication.

This completes the proof of Lemma 4, and thus, the proof of Theorem 2(ii). □

Note that the requirement that the partitions of the set of variables $X$ are balanced is not crucial. The same argument works also in the case when the partitions are only "almost balanced" in a sense that each player gets access to at least a $\lambda$ fraction of all variables, for some $0 < \lambda(n) \leq 1/2$. The proofs of Lemmas 2 and 4 remain the same with only one difference: this time we consider the two cases depending on whether $m \leq \lambda n - 1$ or not. Since each player can see at most $(1-\lambda)n$ rows, we have that in the first case each player will see at least $n - (1-\lambda)n - m \geq 1$ rows. The rest of the proof is the same. The obtained lower bounds on $R(f)$ and $D(g)$ are then of the form $\Omega(\lambda n)$.


## Acknowledgments

I would like to thank Martin Sauerhoff for helpful comments on a prior draft and Georg Schnitger for interesting discussions.


# References


[1] Aho, A., Ullman, J. and Yannakakis, M. (1983): On notions of information transfer in VLSI circuits. In *Proc. of 15th Ann. ACM Symp. on the Theory of Computing*, 133–139.

[2] Babai, L., Frankl, P. and Simon, J. (1986): Complexity classes in communication complexity theory. In *Proc. of 27th Ann. IEEE Symp. on Foundations of Comput. Sci.*, 337–347.

[3] Blum, M. and Impagliazzo, R. (1987): Generic oracles and oracle classes. In *Proc. of 28th Ann. IEEE Symp. on Foundations of Comput. Sci.*, 118–126.

[4] Ďuriš, P., Hromkovič, J., Jukna, S., Sauerhoff, M. and Schnitger, G. (2001): On multipartition communication complexity. In *Proc. of 18th STACS*, Springer Lecture Notes in Computer Science, vol. 2010, 206–217. Journal version in *Information and Computation* (to appear).

[5] Fürer, M. (1987): The power of randomness for communication complexity. In *Proc. of 19th Ann. ACM Symp. on the Theory of Computing*, 178–181.

[6] Halstenberg, B. and Reischuk, R. (1993): Different modes of computation, *SIAM J. Computing* **22**:5, 913–934.

[7] Hartmanis, J. and Hemachandra, L. A. (1987): One-way functions, robustness and non-isomorphism of NP-complete classes. Tech. Rep. DCS TR86-796, Cornell University, 1987.

[8] Hromkovič, J. (1997): *Communication Complexity and Parallel Computing*, Springer-Verlag.

[9] Jayram, T. S., Kumar, R. and Sivakumar, D. (2003): Two applications of information complexity. In *Proc. of 35th Ann. ACM Symp. on the Theory of Computing*, 673–682.

[10] Jukna, S. and Schnitger, G. (2002): Triangle-freeness is hard to detect, *Combinatorics, Probability & Computing* **11**, 549–569.

[11] Jukna, S., Razborov, A., Savický, P. and Wegener, I. (1999): On P versus NP∩co-NP for decision trees and read-once branching programs, *Computational Complexity* **8**:4, 357–370.

[12] Kalyanasundaram, B. and Schnitger, G. (1987): The probabilistic communication complexity of set intersection. In: *Proc. of 2nd Structure in Complexity Theory*, 41–49. Journal version in: *SIAM J. Discrete Mathematics* **5**:4 (1992), 545–557.

[13] Kushilevitz, E. and Nisan, N. (1997): *Communication Complexity*. Cambridge University Press.

[14] Lovász, L. and Saks, M. (1993): Lattices, Möbius functions and communication complexity, *J. Compu. Syst. Sci.* **47**, 322–349.

[15] Nisan, N. (1991): CREW PRAMs and decision trees, *SIAM Journal on Computing* **20**:6, 999–1007.

[16] Papadimitriou Ch. H. and Sipser M. (1982): Communication complexity. In *Proc. of 14th Ann. ACM Symp. on the Theory of Computing*, pp. 196–200. Journal version in: *J. Comput. Syst. Sci.*, **28**:2 (1984), 260–269.

[17] Razborov, A. (1990): On the distributional complexity of disjointness. In: *Proc. of 17th International Colloquium on Automata, Languages and Programming*, Springer Lecture Notes in Computer Science **443**, 249–253. Journal version in: *Theoretical Computer Science* **106**:2 (1992), 385–390.

[18] Sauerhoff, M. (2003): Randomness versus nondeterminism for read-once and read-$k$ branching programs. In *Proc. of 20th Symposium on Theoretical Aspects in Computer Science*, Springer Lecture Notes in Computer Science**2607**, 307–318.

[19] Tardos, G. (1989) Query complexity, or why is it difficult to separate $\mathrm{NP}^A \cap \mathrm{co\text{-}NP}^A$ from $\mathrm{P}^A$ by a random oracle $A$? *Combinatorica* **9**, 385-392.

[20] Yao, A. C. (1979): Some complexity questions related to distributed computing. In: *Proc. of 11th Ann. ACM Symp. on the Theory of Computing*, 209–213.

[21] Yao, A. C. (1983): Lower bounds by probabilistic arguments. In *Proc. of 24th Ann. IEEE Symp. on Foundations of Comput. Sci.*, 420–428.



ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'