Multi-Prover Encoding Schemes and Three-Prover Proof Systems

Gábor Tardos*
Mathematical Research Institute of the
Hungarian Academy of Sciences
Pf: 127, Budapest, H-1364 Hungary
e-mail: tardos@cs.elte.hu

Abstract

Suppose two provers agree in a polynomial p and want to reveal a single value $y = p(x)$ to a verifier where x is chosen arbitrarily by the verifier. Whereas honest provers should be able to agree on any polynomial p the verifier wants to be sure that with any (cheating) pair of provers the value y he receives is a polynomial function of x. We formalize this question and introduce multi-prover (quasi-)encoding schemes to solve it.

Using multi-prover quasi-encoding schemes we are able to develop new results about interactive proofs. The best previous result appears in [BGLR] and states the existence of one-round four-prover interactive proof systems for the languages in NP achieving any constant error probability with $O(\log n)$ random bits and $poly(\log\log n)$ answer size. We improve this result in two respects. First we decrease the number of provers to three, and then we decrease the answer-size to a constant. Using unrelated (parallel repetition) techniques the same was independently and simultaneously achieved by [FK] with only two provers. When the error-probability is required to approach zero, our technique is more efficient in the number of random bits and in the answer size. Showing the fast progress in this central topic of theoretical computer science in the short time since these results were achieved Raz's proof of the parallel repetition conjecture [R] lead to further improvements in the parameters of interactive proofs for NP problems.

1 Introduction

It was established in the past few years that there is a wide ranging and deep connection between multi-prover interactive proofs and transparent proofs on the one hand, and the hardness of approximation on the other. It seems that any progress in the first area clears the way for new applications in the other (cf. the surveys [J], [B]).

Two major results in the first area assert that (i) every NP-language has transparent proofs verifiable with confidence $1-\epsilon$ using $r = O(\log n + |\log(\epsilon)|)$ random bits via $O(|\log(\epsilon)|)$ bit-queries (Arora, Lund, Motwani, Sudan, Szegedy [ALMSS], Arora, Safra [AS]); (ii) every NP-language has one-round interactive proofs with confidence $1-\epsilon$ with a bounded number p of provers, where the verifier uses $r = O(\log n \cdot |\log \epsilon|)$ random bits and the answer of each prover has length $\leq a$. The values of the parameters p and a are critical for the applications. Lapidot, Shamir [LS] obtained this result with $p = 4$ provers and $a = poly(\log n, \log \epsilon)$ answer size; Feige, Lovász [FL] reduced the number of provers

*The author was visiting the Automation and Computation Institute of the Hungarian Academy of Sciences, DIMACS Center and the University of Toronto while part of this research was done. The author is partially supported by NSF grants CCR-92-00788, CCR-95-03254 and the (Hungarian) National Scientific Research Fund (OTKA) grant F 014919

to $p = 2$ while retaining the polylogarithmic answer size. Bellare, Goldwasser, Lund, Russell [BGLR] reduced the answer size to $a = poly(\log\log n, \log\epsilon)$ while requiring $p = 4$ provers.

The following folklore conjecture on the ultimate interactive proof combines the strongest aspects of the results (i) and (ii).

Conjecture. *Every NP-language has one-round interactive proofs with confidence* $1 - \epsilon$ *with* $p = 2$ *provers, where the verifier uses* $r = O(\log n + |\log\epsilon|)$ *random bits and the answer of each prover has length* $a = O(|\log\epsilon|)$.

We achieve the same with $p = 3$ provers for fixed confidence (Theorem 10). Independently Feige and Kilian [FK] achieved this with two provers using a form of parallel repetition. In case the confidence parameter ϵ goes to zero our technique uses both fewer random bits ($O(\log n|\log\epsilon|)$) and smaller answer size ($O(\log^3\epsilon)$) than theirs ($\log n\, poly(1/\epsilon)$ and $poly(1/\epsilon)$). The best previous result was in [BGLR] that used four provers and had answer size (for fixed confidence) $poly(\log\log n)$.

Subsequent to our result Raz [R] proved the parallel repetition conjecture and that lead to one-round interactive proofs for NP languages with $p = 2$ provers, $a = O(|\log\epsilon|)$ answer size where the verifier uses $r = O(\log n|\log\epsilon|)$ random bits to achieve confidence $1 - \epsilon$.

Although Raz's result improves upon the parameters of the MIP's in this paper the totally unrelated proof techniques here may call for interest. We also consider *multi-prover encoding schemes* (MES) introduced in this paper being of independent interest. Parallel repetition techniques seem to be harder to apply for them. Very roughly MES can be thought of as the *encoded theorems* version of two-prover interactive proofs.

In a MIP for a language L the honest provers encode a "proof" for the statement "$x \in L$". A MIP is required to encode a single bit of information about the proof, namely its validity.

In contrast MES provers encode a function from a given family and the verifier must be able to evaluate the function at the value (checkpoint) of his choice. The main point here is that even with cheating provers the answer as a function of the checkpoint must be in the family. In other words the provers must evaluate the *same* function no matter what value the verifier is asking for.

A preliminary version of this paper appeared in the Proceedings of the 9th Annual Structure in Complexity Theory Conference [T].

2 Notation

The input (known both to the provers and the verifier of an interactive proof or encoding scheme) is usually denoted by x. We reserve n to denote the length of x. Below we list the other parameters occurring in this paper.

- The parameters of multi-prover interactive proofs (MIP's) and multi-prover (quasi-)encoding schemes (MES's, MQS's) (Section 3) are:

  r number of random bits used by the verifier

  p number of provers

  q length of the queries to the provers (question-size)

  a length of the provers' answers (answer-size)

  k confidence parameter, the error probability must be $< \epsilon = 2^{-k}$

- The parameters of the function families (defined in Section 3) are:

  t length of the checkpoint T

  z length of the output Z

  y length of the index of a function

- In Section 4 we shall use polynomials over a finite field F to construct interactive proofs and encoding schemes. These polynomials have the following parameters:

  d number of variables (dimension of domain)

  ℓ bound on the total degree (note that for convenience we deal with polynomials of total degree strictly less than ℓ, thus $\ell = 2$ corresponds to the linear case)

  m number of polynomials the verifier needs to evaluate simultaneously (Section 5)

All these parameters and even the size of the field F are functions of the input x. We suppose that all of them are

— positive integers

— polynomially bounded in n

— polynomial time computable

The error probability $\epsilon = 2^{-k}$ may depend on the input. However keeping it a constant ensures that the number of random bits used is $O(\log n)$ which is crucial for all NP-hardness applications.

We use Σ to denote the binary alphabet $\{0, 1\}$ throughout this paper.

A function $E : \Sigma^* \to \Sigma^*$ is a *good encoding function* if it is

— polynomial time computable

— $E(\Sigma^n) \subseteq \Sigma^{cn}$ for some absolute constant c

— for any $x_1 \neq x_2 \in \Sigma^n$ we have $\Delta(E(x_1), E(x_2)) > \delta$ where Δ is the normalized Hamming distance and $\delta > 0$ is an absolute constant.

Good encoding functions are known to exist (cf. [MS, Chapter 10.11]).

3 Definition of encoding schemes

We recall the definition of the single round multi-prover interactive proof systems of Ben-Or, Goldwasser, Kilian, and Wigderson [BGKW]. We use the notation of [BGLR]. A MIP consists of p provers and a verifier. Each prover is a function from questions to answers: $P_i : \Sigma^q \to \Sigma^a$ where q is the question-size and a the answer-size. The verifier is a polynomial time machine receiving the input x and a random string from Σ^r. It produces "questions" $Q_1, \ldots, Q_p$ from Σ^q then it receives the "answers" $P_i(Q_i)$. Finally it accepts or rejects. We say that this proof system accepts a language $L \subseteq \Sigma^*$ with error probability ϵ if

(1) *(completeness)* If $x \in L$ then there exists a set a provers making the verifier surely accept.

(2) *(soundness)* If $x \notin L$ then no set set of provers makes the verifier accept with probability above ϵ.

Let $MIP_1(r, p, a, q, \epsilon)$ stand for the set of languages accepted by such a proof system.

Multi-prover encoding schemes (MES) generalize interactive proofs. Thinking of the following example (similar to the one we use in the proof of Theorem 8) may help to understand the definition of the MES. Suppose the provers encode a univariate polynomial g. Here the input x can determine the field F and the degree ℓ. The verifier must be able to evaluate the function g at any point T of the field with the help of a single question to each honest prover. He wants to be sure that even with cheating provers the answer he is getting (when not catching the provers) is a degree ℓ polynomial of T.

Definition: By *function family* $\mathcal{F}$ we mean a collection $\mathcal{F}_x$ of functions $g : \Sigma^t \to \Sigma^z$ for all strings $x \in \Sigma^*$. Here the parameters $t = t(x)$ and $z = z(x)$ depend on x. We call a function family *polynomial*

if the functions in $\mathcal{F}_x$ can be indexed by the strings Σ^y such that from x, the index of the function $g \in \mathcal{F}_x$ and $T \in \Sigma^t$ the value $g(T)$ is polynomial time computable, and furthermore all the parameters $t = t(x)$, $z = z(x)$, $y = y(x)$ are polynomially bounded and polynomial time computable.

A *multi-prover encoding scheme* (MES) consists of p provers and a verifier. The provers are functions $P_1, \ldots, P_p : \Sigma^q \to \Sigma^a$ just as in a MIP. The provers "see" x and the function $g \in \mathcal{F}_x$ they want to encode (the functions P_i depend on them) but not T. The verifier is polynomial time machine reading a checkpoint $T \in \Sigma^t$ in addition to the input x and the random string $R \in \Sigma^r$. It produces the p queries from Σ^q and then receives the answers from the provers. Finally it produces an output $Z = Z(x, R, T, P_1 \ldots, P_p) \in \Sigma^z \cup \{\text{reject}\}$.

We say that the protocol described is an ϵ-error MES for the function-family $\mathcal{F}$ if

(i) *(completeness)* for all x and $g \in \mathcal{F}_x$ there exist provers $P_1, \ldots P_p$ such that for all checkpoints $T \in \Sigma^t$ and random strings $R \in \Sigma^r$ we have

$$Z(x, R, T, P_1, \ldots, P_p) = g(T)$$

(ii) *(MES soundness)* for all x and provers $P_1, \ldots, P_p$ there exists a function $g \in \mathcal{F}_x$ such that for all checkpoints $T \in \Sigma^t$

$$Prob_{R \in \Sigma^r}\big(Z(x, R, T, P_1, \ldots, P_p) \notin \{g(T), \text{reject}\}\big) < \epsilon$$

Note that soundness means that we can interpret any set of provers as a (perhaps imperfect) attempt to encode a specific function in the family. We call the process of finding that function the *decoding*. Decoding is more obvious in the following definition.

A *multi-prover quasi-encoding scheme* (MQS) for the function family $\mathcal{F}$ with parameters q, a, r, ϵ has the same structure as a MES and it satisfies the same completeness criterion (i) but the following different soundness criterion (ii′):

(ii′) *(MQS soundness)* For all x and all last provers P_p there is a (decoding) function $g : \Sigma^r \to \mathcal{F}_x$ such that for all sets of provers $P_1, \ldots, P_{p-1}$ and all checkpoints $T \in \Sigma^t$

$$Prob_{R \in \Sigma^r}\big(Z(x, R, T, P_1, \ldots, P_p) \notin \{g(R)(T), \text{reject}\}\big) < \epsilon$$

The above definition of the MQS is neither stronger nor weaker than that of the MES. At an MQS the decoding depends on the random string R (this is similar to the definition of a quasi-oracle in [LS]). But the decoding does not depend on the first $p-1$ provers.

Below all MQS have $p = 2$ provers unless otherwise stated.

4 Existence of the encoding schemes

The main result of this section is a scaled down version of [FL] to achieve a MQS for any polynomial function family $\mathcal{F}$. We start with a transparent-proof-like version of MES (Lemma 1). Take any polynomial function family $\mathcal{F}$. Let the parameters of $\mathcal{F}$ be y, t, and z. Take a good encoding function E. We define a "transparent encoding" of the functions in $\mathcal{F}_x$. For a function $g \in \mathcal{F}_x$ with index Y we define $E^*(g)$ to consist of $Y' = E(Y)$ and a transparent proof for each $T \in \Sigma^t$ for the fact that "Y' encodes the index of a function mapping T to Z_T", where Z_T is the correct value of $g(T)$. We do not encode T and $Z = Z_T$ as the verifier receives them as part of its input. The length of $E^*(g)$ is $y^* = 2^t poly(n)$.

Lemma 1. *There exists the following type of a polynomial time verifier V. On input (x, T, Z) V uses $O(\log n)$ random bits, decides which $O(1)$ bits to read from a string $Y^* \in \Sigma^{y^*}$. This choice does not depend on Z. After reading the selected bits it accepts or rejects. Furthermore V satisfies*

(a) *if* $Y^* = E^*(g)$ *for some* $g \in \mathcal{F}_x$ *with* $Z = g(T)$ *then* V *accepts with probability* 1*;*

(b) *for any* x *and* $Y^* \in \Sigma^{y^*}$ *there is a* $g \in \mathcal{F}_x$ *such that for all* T *and all* Z *with* $g(T) \neq Z$ *the probability of acceptance is* $< 1/2$

The proof is a straightforward application of [ALMSS] using the *encoded theorems* model of [BFLS]. This version is implicit [ALMSS] and can be explicitly found in the survey [B] (Theorem 5.6) or in Polishchuk and Spielman's paper [PS] (section 10).

Lemma 2. *Let* $\mathcal{F}$ *be a polynomial function family with parameters* y*,* t *and* z *and let* k *be any parameter. Then there exists a MES for* $\mathcal{F}$ *with* $z + O(k)$ *provers, using* $O(k \log n)$ *random bits. The question-size is* $t + O(\log n)$*, the answer-size is* 1*, and the error probability is at most* 2^{-k}*.*

Proof: We start with the transparent encoding $E^*(g)$ of g in Lemma 1. We use the first z provers to tell the verifier $Z = g(T)$. Following the standard techniques of [FRS] (for valid proof, see [BFL]) we can replace the constant number of queries to $E^*(g)$ with a constant number of provers. The number of random bits used is still $O(\log n)$ and the error-probability increases to a constant below one. $O(k)$ parallel repetition (with different set of provers) gives the lemma. (Naturally, we do not repeat the first z provers.) ♠

In Lemma 4 we deviate slightly from the techniques of [FL, LS]. We shall need the following technical definition and lemma:

Definition: Let F be a finite field, $f : F \to F$ an arbitrary function, $x \in F$ and $\ell \geq 0$ an integer. We define $f_{x\ell}$ the *best* $\leq \ell$ *degree approximation for* f *at* x to be the polynomial that coincides with f for the most values $x' \in F$ among all $\leq \ell$ degree polynomials e satisfying $e(x) = f(x)$. We break ties arbitrarily.

Lemma 3. *For any function* $f : F \to F$ *and any polynomial* e *of degree* $\leq \ell$ *we have*

$$Prob_{x \in F}\big(e(x) = f(x) \text{ but } e \neq f_{x\ell}\big) < \sqrt{2\ell/|F|}$$

Proof: The proof is a simple counting argument using that two different polynomials of degree $\leq \ell$ cannot agree on more than ℓ values. Let $a = |\{x \in F | f(x) = e(x)\}|$. Let S be the set of all $\leq \ell$ degree polynomials agreeing with f on at least a different values. We denote the number of polynomials in S agreeing with f at a value $x \in F$ by $g(x)$. We have $\sum_x g(x) \geq |S|a$. Let us take two different polynomials from S and a value $x \in F$. We count the number of times the polynomials agree on x. This is at most $\binom{|S|}{2}\ell$ as two polynomials from S agree on at most ℓ values. On the other hand if we choose x first we can choose the two polynomials in $\binom{g(x)}{2}$ different ways to ensure both agree with f at x, therefore

$$\binom{|S|}{2}\ell \geq \sum_x \binom{g(x)}{2}$$

If $a < \sqrt{2|F|\ell}$ then the statement of the lemma is trivial. Elementary calculation gives that if on the other hand $a \geq \sqrt{2|F|\ell}$ then $|S| < \sqrt{2|F|/\ell}$. As any polynomial in S other than e agrees with e on at most ℓ values, therefore the number of different values of x satisfying the condition in the lemma is at most $\ell(|S| - 1) < \sqrt{2|F|\ell}$ so the statement of the lemma is true again. ♠

For any question-size q let us take a field F, a dimension d, a polynomial time computable (injective) function $h : \Sigma^q \to F^d$ and a degree ℓ such that the following condition holds:

(*) For any function $P : \Sigma^q \to F$ there is a polynomial $f : F^d \to F$ of degree $< \ell$ such that for any $Q \in \Sigma^q$ we have $P(Q) = f(h(Q))$.

Lemma 4. *Let* $\mathcal{F}$ *be a function family. Suppose there is a MES for* $\mathcal{F}$ *with* p *provers,* r *random bits,* q *question-size, and* ϵ *error. Suppose that a choice of* F*,* d*,* h *and* ℓ *satisfy (*). Suppose that*

the answer-size in the MES for $\mathcal{F}$ is $a < \log|F|$. Then there is a MQS for $\mathcal{F}$ with 2 provers, using $r + pd\log|F|$ random bits, asking questions of size $2pd\log|F|$, receiving an answer of size $p\ell\log|F|$ from the first prover and an answer of size $p\log|F|$ from the second prover and achieving error probability $< \epsilon + p\sqrt{2\ell/|F|}$.

Our protocol follows the structure of the protocols in [LS] and [FL]. In these protocols (and in ours) the verifier chooses a random p-tuple R_V of points in F^d to be the query it sends to the second prover and uses additional randomness to simulate the MES verifier and produce the query to the first prover. The major difference is in the proof of *soundness*. For the proof we have to *decode* the provers' strategy through finding MES-provers that our MQS-provers are close to. These MES-provers may depend on the choice of R_V. [FL] defined the "decoded" prover P_i at a question $Q_i \in \Sigma^q$ using both prover's response when the verifier behaves randomly conditioned on the given choice of R_V and Q_i. We are not able to do this here as the distribution of the queries depends on T and our definition of P_i must not depend on the checkpoint. Therefore we use the second prover's response to queries very close to R_V to define the decoded P_i. This has the added advantage of making the decoding depend only on the second prover's strategy. We have built this advantage into the definition of the MQS.

Proof: We describe the verifier. In brackets we tell what the honest provers should do to encode $g \in \mathcal{F}_x$.

[The provers consider the strategies of the honest provers of the MES encoding of g. These are functions $P_i : \Sigma^q \to \Sigma^a$ for $i = 1, \ldots, p$. We suppose $\Sigma^a \subseteq F$. They consider $< \ell$ degree polynomials $f_i : F^d \to F$ satisfying $P_i(Q) = f_i(h(Q))$ for all $Q \in \Sigma^q$.]

The verifier produces the p queries $Q_1, \ldots, Q_p$ asked by the MES-verifier. He also produces the same number of random points $R_V = (V_1, \ldots, V_p)$ in F^d. He sends a canonical representation of the lines $L_i : F \to F^d$ through $h(Q_i)$ and V_i to the first prover and the points V_i to the second prover.

[The first prover's response is the set of the univariate polynomials $f_i^* = f_i(L_i)$. The second prover's response is the set of values $v_i = f_i(V_i)$.]

The verifier finds values x_{i1} such that $L_i(x_{i1}) = V_i$ and checks if $v_i = f_i^*(x_{i1})$, outputs reject and halts if one of these consistency checks fails. Otherwise it finds values x_{i0} with $L_i(x_{i0}) = h(Q_i)$ and uses the values $f_i^*(x_{i0})$ as answers from the p MES-provers and outputs what the MES-verifier outputs.

Completeness of this protocol is now clear. It is also easy to check that the parameters claimed in the lemma are correct.

In the rest of the proof we prove soundness (condition (ii′)). Let us fix the second prover. We define functions $P_i : (F^d)^p \times \Sigma^q \to \Sigma^a$ for $i = 1, \ldots, p$. Let us take $R_V = (V_1, \ldots, V_p) \in (F^d)^p$ and $Q_i \in \Sigma^q$. Let L_i be the line through $h(Q_i)$ and V_i with x_{i0} and x_{i1} in F such that $L_i(x_{i0}) = h(Q_i)$ and $L_i(x_{i1}) = V_i$. For $u \in F$ let us get $R_V(u)$ from R_V by replacing the ith coordinate V_i by $L_i(u)$. Let the function $f : F \to F$ be defined by $f(u)$ being the ith value in the second prover's answer for the question $R_V(u)$. Now take $P_i'(R_V, Q_i)$ to be the best $< \ell$ degree approximation of f at x_{1i} and let $P_i(R_V, Q_i) = P_i'(R_V, Q_i)(x_{i0})$.

Consider the provers $P_i(R_V, Q_i)$ for any fixed set of points R_V. Using the soundness for the MES we started with there is a function $g = g(R_V) \in \mathcal{F}_x$ such that for every $T \in \Sigma^t$ the verifier in the MES confronted with these provers and T will output something different from $g(T)$ and reject with probability less than ϵ. The decoded function g we obtained here depends on R_V which is part of the random string used, and on the second prover as required.

In our protocol error can come from two sources. Either the simulated MES makes the error, or the simulation fails, that is the first prover's response deviates from $P_1'(R_V, Q_1), \ldots, P_p'(R_V, Q_p)$ what or "decoding" gave. We have just bounded the probability of the first and Lemma 5 (below) bounds the probability of the second type of error. This proves the soundness. ♠

Lemma 5. *For any x, T and any two provers in the protocol in Lemma 4 the probability of the first prover's answer being different from $P_1'(R_V, Q_1), \ldots, P_p'(R_V, Q_p)$ without the verifier catching the provers in the consistency checks is less than $p\sqrt{2\ell/|F|}$.*

Proof: We prove that the probability for a given index i that the ith part of the first prover's response is different from $P'_i(R_V, Q_i)$ without being caught in the consistency check is less than $\sqrt{2\ell/|F|}$. The lemma follows from summing for all i.

We break down the probability space according to all the questions Q_j $(j = 1, \ldots, p)$ the points V_j for $j \neq i$ and even according to the line L_i going through $h(Q_i)$ and V_i. We prove that conditioned on any combination of values for these objects the conditional probability of the ith part of the first prover's response deviating from $P'_i(R_V, Q_i)$ without being caught is less than $\sqrt{2\ell/|F|}$. This of course implies the same bound for the total probability.

After all the conditions the only thing random is V_i which is a random point on the line L_i. The first prover's answer is now fixed. Let as call the ith part of it e. Let us call $f(u)$ the ith part of the second prover's answer when $V_i = L_i(u)$. The consistency test is passed if $f(u) = e(u)$ with the random value $u = L_i^{-1}(V_i)$. We have the deviation in the ith coordinate if e is not the best $< \ell$ degree approximation of f at u. Lemma 3 bounds the probability of these two things happening together. ♠

Here we state what Lemma 4 gives when F, d, h, and ℓ are chosen the simplest way, i. e. when h is the identity.

Theorem 6. *Let $\mathcal{F}$ be a polynomial function family with the parameters $y \leq poly(n)$, $t \leq poly(\log n)$, $z \leq poly(\log n)$ and let $k \leq poly(\log n)$. Then there exists a 2-prover MQS for $\mathcal{F}$ using $poly(\log n)$ random bits, with question- and answer-sizes $poly(\log n)$ achieving error probability 2^{-k}.*

Proof: We start with Lemma 2 and get a MES using $p = poly(\log n)$ provers, $poly(\log n)$ randomness, the question-size is $q = poly(\log n)$, answer-size is 1 and error probability $< 2^{-k-1}$. Let $d = q$, $\ell = d+1$, and $|F| > 2^{2k+3}\ell p^2$ (but, say smaller than twice that). We can chose h to be the identity as any function $\Sigma^d \to F$ has a multilinear extension $F^d \to F$, so (*) is satisfied. The error probability of the MQS given by Lemma 4 is $< 2^{-k-1} + p\sqrt{2\ell/|F|} < 2^{-k}$. It is easy to check that all other parameters of the two-prover MQS are $poly(\log n)$. ♠

5 Three-prover proof systems

In this section we give efficient three-prover proof systems for NP with $poly(\log\log n)$ answer size (Theorem 8). The nice and simple idea of this proof will help to understand the constant answer size proof systems (Theorem 10) in the next section.

We start with an overview of the proof. The straightforward modification of [FL] (scaling down from NEXPTIME to NP and changing multilinear encoding to multi-lowdegree encoding as in [BFLS]) gives Lemma 7, a two-prover proof system for NP using $O(\log n)$ random bits and $\log^2 n$ answer-size to achieve any constant error probability. This appears in [BGLR] in detail. In fact taking a closer look one may realize that one of the provers gives a very short $O(\log\log n)$ long answer. The length of the other prover's answer is $\log^2 n$. (Or for that matter this length can be made $\log^c n$ for any $c > 1$ by making $h = max(k, (c-1)\log\log n)$ in the lemma, but that is still too long.) In fact this answer contains a constant number of polynomials of degree $\log^2 n/\log\log n$ over a field F of size $|F| = poly(\log n)$. The verifier uses two values of each of these polynomials in deciding whether to accept or to reject. (It is important here that the verifier knows which values it will use before it receives the answers.) This makes it possible to replace this prover by two provers providing a quasi-encoding scheme for this answer. This is an [AS]-type recursion, increasing the number of provers to 3.

Lemma 7. *[BGLR] Let $k = O(\log n)$ and $h = max(k, \log\log n)$. Then there are parameters $r = O(k\log n)$, $a = O(k\log n 2^h)$ and $q = O(k\log n)$ such that $NP \subseteq MIP_1(r, 2, a, q, 2^{-k})$.*

We shall need further details from the protocol proving this. There is a field F of size $2^{O(h)}$, the first prover gives $O(k)$ polynomials over F of degree $O(2^h/h\log n)$ the second prover gives $O(k)$ values. The verifier first extracts one value of each polynomial and compares them to the values the second prover gave. If they differ he rejects. Otherwise he extracts another value of each polynomial checks if they

are 0 and uses these $O(k)$ bits as input of an $O(k)$ size Boolean circuit, accepts if the circuit computes 1.

Theorem 8. *For any $k \leq poly(\log\log n)$ there are parameters $r = O(k \log n)$, $q = O(k \log n)$, and $a = poly(\log\log n)$ such that $NP \subseteq MIP_1(r, 3, a, q, 2^{-k})$.*

Proof: Take any language $L \in NP$. We are going to refer to the MIP protocol of Lemma 7 (with $k+1$ in place of k) for L as the original protocol and design a new MIP protocol with three provers. The only problem with the original protocol is the first prover's long answer. This answer consists of $m = O(k)$ polynomials. The verifier uses two values per polynomial only, so it is a natural idea to encode the answer using MQS for the following function family $\mathcal{F}$.

- The input $\hat{x}$ of size $\hat{n}$ contains the size of a field F, $|F| \leq poly(\hat{n})$, a number $m = O(\log \hat{n})$ and $\ell = poly(\hat{n})$ (and lots of padding).
- A function $g \in \mathcal{F}_{\hat{x}}$ is indexed by an m-tuple of polynomials $(Y_1, \ldots, Y_m)$ of degree $< \ell$ over F.
- A checkpoint T consists of the values $T_{ij} \in F$ for $i = 1, \ldots, m$, $j = 0, 1$.
- The output $Z = g(T)$ contains $Z_{ij} = Y_i(T_{ij})$ for $i = 1, \ldots, m$, $j = 0, 1$.

This function family $\mathcal{F}$ is clearly polynomial, so by Theorem 6 there exists an MQS for it with all its parameters being $poly(\log \hat{n})$ achieving confidence $1 - 2^{-\hat{k}}$ with any $\hat{k} = poly(\log \hat{n})$.

Now we transform the original protocol. The second prover remains intact. We replace the first prover by two provers say provers A and B. Here is how the verifier works. In brackets we give the honest provers' strategy.

The verifier computes the questions Q_1 for the first prover and Q_2 for the second provers in the original protocol. He sends Q_2 to the second prover but as the first prover does not take part in the protocol the verifier has to work some more. He also computes the points where the original verifier would evaluate the $m = O(k)$ polynomials of degree $< \ell$ in the first prover's answer. These 2 points per polynomial constitutes T. Now he takes the MQS protocol for $\mathcal{F}$, finds (a canonical) $\hat{x}$ of size $\hat{n} = \log n$ describing the correct F, m and ℓ and computes the two queries Q_A and Q_B according to the MQS with confidence parameter $\hat{k} = k + 1$. He sends Q_1 and Q_A to prover A and Q_1 and Q_B to prover B.

[The second prover responds as in the original protocol. Provers A and B compute $\hat{x}$ and behave as the honest provers behave in the MQS for $\mathcal{F}$ on input $\hat{x}$ when encoding the function indexed by the first prover's answer to Q_1.]

The verifier uses the answers of prover A and B to simulate the MQS-verifier. If it outputs reject then he also rejects. Otherwise he uses the result Z as if it was what the polynomials in the first prover's response evaluate to. He follows the simulation of the original protocol and accepts or rejects as it does.

The completeness is easy to see again. The parameters claimed are also easy to verify. For example the number of random bits used in producing Q_1 and Q_2 is $r_1 = O(k \log n)$ and to produce Q_A and Q_B the verifier uses $r_2 = poly(\log\log n)$ more random bits. The same way the answer-sizes and the size of the additional questions Q_A and Q_B are also $poly(\log\log n)$

The soundness is left to be proven. Suppose $x \notin L$. By the definition of the MQS for any Q_1 there is a function $g_{Q_1} : \Sigma^{r_2} \to \mathcal{F}_{\hat{x}}$ such that for any T the probability for random coinflipps R_2 in the second part $Prob_{R_2}(Z \notin \{g_{Q_1}(R_2)(T), \text{reject}\}) < 2^{-k-1}$. As $g_{Q_1}(R_2) \in \mathcal{F}_{\hat{x}}$ is indexed by a possible answer of the first prover, for any fixed R_2 we can consider g to be a function from the questions (Q_1) to the answers of the first prover. This is a strategy for the first prover, thus taking it together with the second prover's strategy they yield acceptance with probability $< 2^{-k-1}$ by the soundness of the original protocol. Acceptance can come from two errors, either $Z = g_{Q_1}(R_2)(T)$ (and the original protocol makes an error), or $Z \notin \{g_{Q_1}(R_2)(T), \text{reject}\}$ (the MQS protocol errs) and the probability of either is less than 2^{-k-1}. So the total probability of an input $x \notin L$ being accepted is $< 2^{-k}$. ♠

6 Constant answer-size

In this section we reduce $poly(\log\log n)$ answer-size of Theorem 8 to a constant (Theorem 10). Our starting point is the MIP in Lemma 7 again. We reduce the answer-size by replacing both provers by a two-prover quasi-encoding scheme each encoding what their answer would be, the same way we replaced one of them in the preceding section. The number of the resulting four provers can be decreased to three by "merging" the first provers of each scheme. This does not cause a problem since the second prover alone is enough for the decoding. This is the only point in this paper where we make use of this feature of an MQS.

There are several problems to overcome to implement the strategy outlined above. First we cannot use an encoding scheme for the same function family $\mathcal{F}$ as in the preceding section as the output of that is several elements of the field F used in the two-prover MIP, each of length $\log\log n$ so we could not hope for shorter answer-size. To overcome this difficulty we use a good encoding function E to encode the output and only ask for a randomly chosen constant number of bits from the result.

Even after reducing the size of the output of the function family we cannot use Theorem 6 to get a constant answer-size MQS. We have to go back to Lemma 4 and find better values of F, d, and ℓ satisfying (*). In order to get constant answer-size $|F|$ and ℓ must be constants. As the total number of random bits must be kept $O(\log n)$ the dimension d has to be $O(\log n)$. (Here n is the size of the input for the MIP, not the size of the input of this function family.) The technique of [ALMSS] (last step of the recursion, robust encoding) is applicable here, yields linear functions ($\ell = 2$) but only allows for an encoding of a witness of length $O(\log n)$. As the first prover's answer in the Lemma 7 MIP is longer, we must use a different technique. A simple observation allows us to encode longer, $poly(\log n)$ length witnesses with constant ℓ, $|F|$ and answer-size (Theorem 9). We remark here that the same trick can be used in the [ALMSS] proof to save one of the three steps of the recursion there.

Theorem 9. *Let $\mathcal{F}$ be a polynomial function family. Let k be a parameter and suppose the parameters of $\mathcal{F}$ satisfy $y \le poly(n)$, $t = O(k \log n)$ and $z = O(k)$. Then there exists a MQS for $\mathcal{F}$ using $O(k^3\sqrt{n})$ random bits, with question-size $O(k^3\sqrt{n})$ and with answer size $O(k^3)$ achieving error probability $< 2^{-k}$.*

Proof: We start with Lemma 2 and get a MES for $\mathcal{F}$ with $p = O(k)$ provers, $r = O(k \log n)$ random bits, the question-size is $q = O(k \log n)$, the answer-size is 1, and the error probability is $< 2^{-k-1}$. The choice of F, d, ℓ and h in Theorem 6 does not suffice here, we have to choose them differently. First and foremost we want to keep ℓ down, but we also want to keep the number of random bits under control.

Let $s = q/\log n$. We take $d = 2s\sqrt{n}$, $\ell = 2s + 1$, $|F| > 2^{2k+3}\ell p^2$ (but at most twice as much). Let us choose $h : \Sigma^q \to F^d$ to be polynomial time computable injective function such that its image contains only points with $2s$ coordinates 1 and the rest 0. Such function exists as we were careful enough to ensure $\binom{d}{2s} > 2^q$. There exists a degree $2s$ monomial for any point in $h(\Sigma^q)$ making it 1 and making all other point in $h(\Sigma^q)$ 0, so any function on $h(\Sigma^q)$ can be extended to F^d to a polynomial of degree $2s$. This makes our choice of h and ℓ satisfy the condition they have to for Lemma 4 to apply. It is easy to check that the parameters Lemma 4 gives are the ones we claimed in the theorem. ♠

Theorem 10. *Let $k = O(\log\log n)$ be a parameter. Then there are parameters $r = O(k \log n)$, $q = O(k \log n)$ and $a = O(k^3)$ such that $NP \subseteq MIP_1(r, 3, a, q, 2^{-k})$.*

We remark here that the proof of the parallel repetition theorem [R] improves this result in two respects. It decreases the number of provers to two and the answer-size to $O(k)$. The conjecture in the introduction calls for further improvement in the number of random bits to $O(k + \log n)$. Such improvement is not likely to be possible via parallel repetition techniques (cf. [FK2]).

Before the formal proof we give an outline and define the function families used in the protocol.

As in the proof of Theorem 8 we are going to use our provers in pairs to encode the two provers' response in the Lemma 7 MIP for the same NP language. The first prover response consists of m polynomials, and the verifier uses two values per polynomial, one for consistency check against the second prover's response, the other as input to evaluate a small Boolean circuit. We must make our function family's output short, therefore it's natural to build the circuit in the checkpoint so the output

of the function family contains only the output of the circuit rather than its input. For the consistency check we use a good encoding function for the strings that should be equal in the two provers' answers and build in the checkpoint a few positions of the encoded string and let the output of the polynomial function families contain the so defined substring only.

Let us fix a good encoding function E. Let the absolute constant c be the expansion of E, i. e. $|E(Y)| = c|Y|$. We start with describing the function families we are going to use in this construction. We use $\mathcal{F}'$ to encode the first prover's response, and $\mathcal{F}''$ to encode the second prover's response. Here we define $\mathcal{F}'$:

- The input $\hat{x}$ of length $\hat{n}$ for $\mathcal{F}'$ contains the size of a field F, $|F| \leq poly(\hat{n})$; and parameters $m = O(\log \hat{n})$, $\ell \leq poly(\hat{n})$, and $z = O(\log \hat{n})$ (and padding).
- A function $g \in \mathcal{F}'_{\hat{x}}$ is indexed by an m-tuple of univariate polynomials $(Y_1, \ldots, Y_m)$ of degree $< \ell$ over F.
- A checkpoint T consists of elements $T_{ij} \in F$ for $i = 1, \ldots, m$, $j = 0, 1$; z bit-positions from $\{1, \ldots, cm \log |F|\}$; and an $O(m)$ size Boolean circuit C on m input variables.
- To compute the output $Z = g(T)$ first compute the concatenation of the values $Y_i(T_{i1}) \in F$. Let us call Z_0 this string of length $m \log |F|$. The first z bits of the output contain the bits from $E(Z_0)$ specified in T. The last bit of the output is the output of the circuit C on the input bits that are the truth values of $Y_i(T_{i0}) = 0$. So, confusingly, the length of the output is $z + 1$.

Let us describe $\mathcal{F}''$ now.

- The input $\hat{x}$ of length $\hat{n}$ for $\mathcal{F}''$ contains two numbers $y_2 \leq poly(\log \hat{n})$ and $z = O(\log \hat{n})$ (and padding).
- The functions $g \in \mathcal{F}''_{\hat{x}}$ are indexed with strings Y of length y_2.
- A checkpoint T contains z bit-positions from $\{1, \ldots, cy_2\}$.
- The output $Z = g(T)$ is the z bits appearing in $E(Y)$ in the positions given by T.

Both $\mathcal{F}'$ and $\mathcal{F}''$ are clearly polynomial.

Proof of Theorem 10: Let $L \in NP$ be an arbitrary language. We are going to show that the following MIP protocol works for L and and has the parameters claimed in the theorem.

We will refer to the MIP protocol for L in Lemma 7 (with error probability 2^{-k-2}) as the original protocol with the first and second provers. Our protocol also uses the efficient MQS for $\mathcal{F}'$ and $\mathcal{F}''$ (with error probability 2^{-k-2}) claimed in Theorem 9.

We call the three provers of our MIP protocol for L prover A, prover $1B$, and prover $2B$. We describe what our verifier does. In brackets we give the honest provers' strategy.

The protocol uses the structure of the original protocol described after Lemma 7. The verifier computes the field F, the degree ℓ and the number m there. It finds an $\hat{x}_1$ of length $\hat{n} = \log n$ describing the correct F, m, ℓ, and a number $z = O(k)$. It finds another $\hat{x}_2$ of the same length describing $y_2 = m \log |F|$ and z.

[As all this computation was based on x alone we may suppose the provers "know" $\hat{x}_1$ and $\hat{x}_2$.]

- The verifier simulates the original verifier and computes the queries Q_1 for the first prover and Q_2 for the second.
- It computes the two places T_{i0} and T_{i1} where the original verifier would evaluate the ith polynomial in the first prover's answer. Let us call T_0 the concatenation of all these places (as strings) for $i = 1, \ldots, m$.
- It also computes the circuit C the the original verifier would use.

- Using new random bits the verifier produces T_2, a sequence of z random elements from $\{1, \ldots, cy_2\}$.
- The verifier then simulates the MQS for $\mathcal{F}'$ on input $\hat{x}_1$ with the checkpoint T_1 being the concatenation of T_0, T_2, and C. It computes the queries Q_{1A} and Q_{1B}
- It also simulates the MQS for $\mathcal{F}''$ on input $\hat{x}_2$ with the checkpoint T_2 and computes the queries Q_{2A} and Q_{2B}
- The verifier sends Q_1, Q_2, Q_{1A}, and Q_{2A} to prover A. It sends Q_1 and Q_{1B} to prover $1B$. Finally it sends Q_2 and Q_{2B} to prover $2B$.

[Provers A and $1B$ both receive Q_1 so they "know" the honest first prover's answer Y_1 in the original protocol. Their answers A_{1A} and A_{1B} are the answers of the honest MQS-provers for $\mathcal{F}'$ encoding the function indexed by Y_1.

The same way provers A and $2B$ can simulate the honest MQS-provers encoding the function from $\mathcal{F}''_{\hat{x}_2}$ indexed by the honest second prover's answer Y_2 to Q_2. Their answers are A_{2A} and A_{2B}.

As prover A participated in both simulation, its final answer is a concatenation of A_{1A} and A_{2A}.]

The verifier uses A_{1A} and A_{1B} to simulate the MQS for $\mathcal{F}'$ and produces an output Z_1. It uses A_{2A} and A_{2B} to simulate the MQS for $\mathcal{F}''$ and produces as output Z_2. It rejects if any of these conditions hold:

- Z_1 or Z_2 is reject,
- the first z digits of Z_1 does not equal to Z_2,
- the last digit of Z_1 is 0.

Otherwise the verifier accepts. This finishes the description of our protocol.

The completeness is easy to see. It is also easy to see that the parameters claimed are correct. For example the random string used by the verifier has four parts: R_0 used to simulate the original protocol, R_t used to generate T_2, the random bit-positions for the consistency check, R_1 used to simulate the MQS for $\mathcal{F}'$, and R_2 to simulate the MQS for $\mathcal{F}''$. Here $|R_0| = O(k \log n)$, $|R_t| = O(k \log k)$, and by Theorem 9 both $r_1 = |R_1|$ and $r_2 = |R_2|$ are $O(k^3\sqrt{\log n})$.

The rest of this proof is the proof of soundness. Suppose the input x is not in L, therefore it should be rejected. Let us fix the three provers. We want to prove that the probability of acceptance is small.

Prover $1B$ is a function taking Q_1 and Q_{1B} and returning A_{1B}. For any fixed Q_1 this is a strategy for the last prover in the MQS for $\mathcal{F}'$. By the soundness of the MQS there is a function $g_{1Q_1} : \Sigma^{r_1} \to \mathcal{F}'_{\hat{x}_1}$ such that

$$Prob(Z_1 \notin \{g_{1Q_1}(R_1)(T_1), \text{reject}\}) < 2^{-k-2}.$$

Similarly the soundness of the other MQS implies the existence of a function $g_{2Q_2} : \Sigma^{r_2} \to \mathcal{F}''_{\hat{x}_2}$ for any Q_2 such that

$$Prob(Z_2 \notin \{g_{2Q_2}(R_2)(T_2), \text{reject}\}) < 2^{-k-2}.$$

Let us fix the random strings R_1 and R_2. The function mapping Q_1 to the index of $g_{1Q_1}(R_1)$ is a strategy for the first prover in the original protocol. Similarly the function mapping Q_2 to the index of $g_{2Q_2}(R_2)$ is a strategy for the second prover. By the soundness of the original protocol these provers make the verifier accept x with probability less than 2^{-k-2}.

The acceptance in our protocol may come for four types of errors. We have just proved that three types of error (error in the original protocol or in one of the MQS's) has probability less than 2^{-k-2} each. The last type of error is when our simulation of the original protocol fails, but not because of an error in one of the MQS's. This can only happen if the polynomials in the (assumed) first prover's answer evaluated at T_{1i} as a string differs from the (assumed) second prover's answer, but the random z bit-positions in their encodings by E agree. As any one bit-position reveals the difference with a positive absolute constant probability by the choice of E, we can choose $z = O(k)$ in such a way that this last kind of error has probability less than 2^{-k-2}.

This makes the total error probability $< 2^{-k}$ and the proof complete. ♠

It is interesting to see that in the protocol described above (the relevant part of) the answer of the Lemma 7 MIP's both provers is contained in a single prover's answer (prover A). So consistency checks in the simulated MIP are performed between two parts of that answer. What makes the protocol still work is that the two parts of that answer is then checked against different provers ($1B$ and $2B$).

7 Further results and open problems

Cubic programming is the problem of maximizing a *real* polynomial $f(x_1, \ldots, x_n)$ of total degree 3 over a compact region defined by linear constraints: $\{x \in \mathrm{R}^n | Ax \leq b\}$. Let f^* be the maximum and f_* be the minimum of f on the feasible region. Here we define $\bar{f}$ to approximate the maximum within a factor of c if $|\bar{f} - f^*| \leq c|f^* - f_*|$. This definition is invariant under shifting f with an additive constant unlike the definition in which we would compare $\bar{f} - f^*$ to f^*, cf. [V]. Using the techniques of [BR, FL] our Theorem 8 implies

Theorem 11. *For any constant $0 < c < 1$ approximating the maximum of a cubic program within c is NP-hard.*

The same holds even for quadratic programs by [FK].

We believe that the concepts MES and MQS are of independent interest. Although we introduced them in this paper mainly as tools to help us build efficient interactive proofs, these encoding schemes (particularly MES) are conceptually clear and simple objects, thus finding efficient encoding schemes may turn out to be useful beyond the one application here. These encoding schemes generalize interactive proofs roughly the way the *encoded theorems* version of [BFLS] generalize transparent proofs.

In the rest of this section we summarize what we know about encoding schemes and what related questions are still open. The first two theorems provide good MQS's for any polynomial function family.

Theorem 12. *Let $\mathcal{F}$ be a polynomial function family, $k = O(\log n)$ a parameter, and $c > 0$ a constant. Suppose $t = O(\log n)$ and $z = O(k)$. Let $h = max(k, c \log\log n)$. Then there is a two-prover MQS for $\mathcal{F}$ using $O(k \log n)$ random bits and query-size and $O(k \log n 2^h)$ answer size with error probability $< 2^{-k}$.*

The proof is a simple application of Lemma 4. The way to satisfy the condition (*) is similar to the technique in the proof of Lemma 7.

Using this theorem instead of Lemma 7 we can extend Theorem 10 to polynomial function families. This is the only time we refer to MQS's with more than two provers.

Theorem 13. *Let $\mathcal{F}$ be a polynomial function family and $k = O(\log\log n)$ a parameter. Suppose $t = O(\log n)$ and $z = poly(k)$. Then there is a three-prover MQS for $\mathcal{F}$ with $O(k \log n)$ random bits and question-size and $poly(k)$ answer size with error probability 2^{-k}.*

The proof is a straightforward modifications of the proof of Theorem 10.

It is an interesting open problem to find the equivalent of these theorems (or even Theorem 6) with MES's in place of MQS's. We introduced the technical definition of MQS to circumvent the problem of finding efficient MES's, but the MES is the conceptually clear and natural version of the encoding scheme. It is frustrating that we are unable to find an efficient solution to the simple problem outlined in the first paragraph of the abstract without relaxing the natural soundness condition.

One can also hope for a protocol that generalizes both MES's and MQS's.

Definition: We call a multi-prover encoding scheme *strong* if the decoding of g in (ii) of the definition of the MES that a-priory may depend on the p provers and the input x in fact depends only on x and the last prover P_p.

The following conjecture claims the existence of the strongest possible MES for any polynomial function family that is possible without a collapse in the complexity classes. It is the MES analogue of the conjecture in the introduction. In fact this conjecture implies the one in the introduction.

Conjecture. *For every polynomial function family g with output size z and for every confidence parameter ϵ there is an ϵ error two-prover strong MES that uses $r = O(|\log \epsilon| + \log n)$ random bits and has $a = O(|\log \epsilon| + z)$ answer size.*

Acknowledgments

The author is grateful to László Babai for helpful discussions and encouragement.

References

[ABSS] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. In: *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, 1993, 724–733.

[ALMSS] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In: *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, 1992, 14–23.

[AS] S. Arora and S. Safra. Probabilistic checking of proofs. In: *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, 1992, 2–13.

[B] L. Babai. Transparent proofs and limits to approximation. In: *Proceedings of the First European Congress of Mathematics*, Birkhäuser, to appear.

[BFLS] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In: *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, 1991, 21–31.

[BFL] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols *Computational Complexity* **1** (1991), 3–40.

[BGKW] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: how to remove intractability assumptions. In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, 1988, 113–131.

[BGLR] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In: *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, 1993, 294–304.

[BR] M. Bellare and P. Rogaway. The complexity of approximating a nonlinear program. n: *Complexity in Numerical Optimization*, P. Pardalos, ed., World Scientific, Singapore 1993.

[FK] U. Feige and J. Kilian, Two prover protocols — low error at affordable rates, *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, 1994.

[FK2] U. Feige and J. Kilian. Impossibility results for recycling Random bits in two-prover proof systems. In: *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, 1995, 457–468

[FL] U. Feige and L. Lovász. Two-prover one-round proof systems: their power and their problems. In: *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, 1992, 733-744.

[FRS] L. Fortnow, J. Rompel, and M. Sipser. On the power of multiprover interactive protocols. In: *Proceedings of the 3rd Structure in Complexity Theory Conference*, IEEE, 1988, 156–161.

[J] D. S. Johnson. The NP-completeness column: an ongoing guide. *Journal of Algorithms* **13** (1992), 502–524.

[LS] D. Lapidot and A. Shamir. Fully parallelized multi prover protocol for NEXPTIME. In: *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science*, 1991, 13–18.

[LY] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. In: *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, 1993, 286–293.

[MS] F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes*, North-Holland, Amsterdam, 1977.

[PS] A. Polishchuk and D. Spielman. Nearly-linear size holographic proofs. In: *Proceedings of the 26 Annual ACM Symposium on Theory of Computing*, 1994, 194–203.

[R] R. Raz. The parallel repetition theorem. In: *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, 1995, 447–456.

[T] G. Tardos. Multi-prover encoding schemes and three-prover proof systems. In: *Proceedings of the 9th Annual Structure in Complexity Theory Conference*, 1994, 308–317.

[V] S. Vavasis. On approximation algorithms for concave programming. *Recent Advances in Global Optimization*, C. A. Floudas and P. M. Pardalos, 3–18, Princeton University Press, 1992.