On the Space Complexity of Linear Programming with Preprocessing

Yael Tauman Kalai[*]
Microsoft Research
yael@microsoft.com

Ran Raz[†]
Weizmann Institute of Science, Israel, and the Institute for Advanced Study, Princeton, NJ
ran.raz.mail@gmail.com

Oded Regev[‡]
Courant Institute of Mathematical Sciences, New York University

ABSTRACT

It is well known that Linear Programming is P-complete, with a log-space reduction. In this work we ask whether Linear Programming remains P-complete, even if the polyhedron (i.e., the set of linear inequality constraints) is a fixed polyhedron, for each input size, and only the objective function is given as input. More formally, we consider the following problem: maximize $c \cdot x$, subject to $Ax \leq b; x \in \mathbb{R}^d$, where A, b are fixed in advance and only c is given as an input.

We start by showing that the problem remains P-complete with a log-space reduction, thus showing that $n^{o(1)}$-space algorithms are unlikely. This result is proved by a direct classical reduction.

We then turn to study *approximation* algorithms and ask what is the best approximation factor that could be obtained by a small space algorithm. Since approximation factors are mostly meaningful when the objective function is non-negative, we restrict ourselves to the case where $x \geq 0$ and $c \geq 0$. We show that (even in this possibly easier case) approximating the value of max $c \cdot x$ (within any polynomial factor) is P-complete with a polylog space reduction, thus showing that $2^{(\log n)^{o(1)}}$-space approximation algorithms are unlikely.

The last result is proved using a recent work of Kalai, Raz, and Rothblum, showing that every language in P has a no-signaling multi-prover interactive proof with poly-logarithmic communication complexity. To the best of our knowledge, our result gives the first space hardness of approximation result proved by a PCP-based argument.

[*]This work was done (in part) while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467.

[†]Research supported by the Israel Science Foundation grant No. 1402/14, by the I-CORE Program of the Planning and Budgeting Committee and the Israel Science Foundation, by the Simons Collaboration on Algorithms and Geometry, by the Fund for Math at IAS, and by the National Science Foundation grant No. CCF-1412958.

[‡]Supported by the Simons Collaboration on Algorithms and Geometry and by the National Science Foundation (NSF) under Grant No. CCF-1320188. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
ITCS'16, January 14–16, 2016, Cambridge, MA, USA.
© 2016 ACM. ISBN 978-1-4503-4057-1/16/01 ...$15.00.
DOI: http://dx.doi.org/10.1145/2840728.2840750.

Categories and Subject Descriptors

G.1.6 [**Optimization**]: Linear programming; F.1.3 [**Complexity Measures and Classes**]: Reducibility and completeness

Keywords

Linear programming, P-completeness, space complexity, preprocessing

1. INTRODUCTION

Linear programs often arise in practice, and algorithms for Linear Programming are widely deployed. There has been a major effort to construct fast algorithms for Linear Programming, resulting with very efficient (polynomial time) algorithms (e.g., [7, 18, 15, 9, 6, 16]).

Recall that a linear program is a constrained optimization problem of the form:

$$\text{maximize } c \cdot x$$
$$\text{subject to } Ax \leq b; x \in \mathbb{R}^d$$

where $c \in \mathbb{R}^d$ and $b \in \mathbb{R}^n$, and A is an $n \times d$ matrix. The vector c is the objective function, and the set $H = \{x : A \cdot x \leq b\}$ is the set of feasible points. If it is non-empty, H is a convex polyhedron.

1.1 The Space Complexity of Linear Programming

A fundamental question is: what is the *space complexity* of Linear Programming?

The space complexity of Linear Programming was first studied in the late 70's. Dobkin, Lipton, and Reiss [8], followed by Serna [23], proved that approximating Linear Programming is P-complete with a log-space reduction, thus showing that $n^{o(1)}$-space algorithms for (approximating) Linear Programming are unlikely. (See also Feige and Kilian's paper [10] for an alternative proof.)

For the special case of positive Linear Programming (where all variables in x, as well as all the coefficients in c, A, b are non-negative), Luby and Nisan [20] gave a polynomial time and $\mathsf{polylog}$-space approximation algorithm.

1.2 Our Results

The above results, by Dobkin, Lipton and Reiss [8] and Serna [23], show that any language $L \in \mathsf{P}$ has a $\mathsf{log\text{-}space}$ reduction that converts any instance $x^* \in \{0,1\}^n$ into a linear program of size $\mathsf{poly}(n)$, such that if $x^* \in L$ then the the maximum value of the objective function on the polyhedron is 1, and if $x^* \notin L$ then the maximum value of the objective function on the polyhedron is 0. The polyhedron of the resulting linear program is not fixed and depends on x^*. In this work we ask whether a similar result can be obtained when the polyhedron is some fixed polyhedron that depends only on the input size and the running time, and doesn't depend on the instance x^* or the language L. A positive answer implies that small-space algorithms for Linear Programming are unlikely, even if one allows a *preprocessing* phase that takes the polyhedron as input and runs in unbounded time and space. Similarly, a positive answer implies that parallel algorithms for this problem running in $\mathsf{polylog}(n)$ time on a polynomial number of processors are unlikely.

Our first result shows that Linear Programming remains P-complete (with a $\mathsf{log\text{-}space}$ reduction) even when the polyhedron is fixed: We show that any language $L \in \mathsf{P}$ has a $\mathsf{log\text{-}space}$ reduction that converts any instance $x^* \in \{0,1\}^n$ into a linear program of size $\mathsf{poly}(n)$, such that if $x^* \in L$ then the the maximum value of the objective function on the polyhedron is at least 1, and if $x^* \notin L$ then the maximum value of the objective function on the polyhedron is at most 0. The polyhedron of the resulting linear program is a *fixed* polyhedron that depends on n but is otherwise independent of x^*,[1] and only the objective function depends on x^*. This shows that (for some polyhedrons) $n^{o(1)}$-space algorithms for Linear Programming are unlikely, even if the polyhedron is fixed and is not part of the input. This result is proved by a direct classical reduction, building on the techniques of [8] and [23].

We then turn to study approximation algorithms and ask what is the best factor of approximation that can be obtained by a small-space algorithm. We note that in the most general case of Linear Programming, any gap (in the value of the program), say a gap between 1 and $1-\varepsilon$, can be easily amplified to a gap between 1 and 0, by applying the linear transformation $f(z) = (z-(1-\varepsilon))/\varepsilon$ on the objective function. Therefore, in order for the approximation question to be meaningful, we restrict ourselves to the *semi-positive* case, where $x \geq 0$ and $c \geq 0$. We believe that in the semi-positive case, gaps cannot be easily amplified, and hence are much harder to establish, and that the semi-positive case is a particularly interesting one from the point of view of hardness of approximation.

Our second result shows that in the semi-positive case (where $x, c \geq 0$), and even when the polyhedron is fixed, approximating Linear Programming is P-complete under a $\mathsf{quasi\text{-}poly}$ time and $\mathsf{polylog}$ space reduction: We show that any language $L \in \mathsf{P}$ has a $\mathsf{quasi\text{-}poly}$ time and $\mathsf{polylog}$ space reduction that converts any instance $x^* \in \{0,1\}^n$ into a linear program of quasi-polynomial size, such that if $x^* \in L$ then the the maximum value of the objective function on the polyhedron is 1, and if $x^* \notin L$ then the maximum value of the objective function on the polyhedron is less than $2^{-\mathsf{polylog}(n)}$. The polyhedron of the resulting linear program is a *fixed* polyhedron that depends on n (and the running time) but is otherwise independent of x^* (and L), and only the objective function depends on x^*. This shows that (for some polyhedrons) $2^{(\log n)^{o(1)}}$-space approximation algorithms for semi-positive Linear Programming (within a quasi-polynomial factor) are unlikely, even if the polyhedron is fixed and is not part of the input.

The proof of this second result uses the recent work of [14], that shows that any language in P has a multi-prover interactive proof with $\mathsf{polylog}$ communication complexity, which is secure against *no-signaling* cheating provers.

We note that our first result can be modified to yield semi-positive instances, however the resulting hardness of approximation factors are miniscule. We also note that if in addition to $x, c \geq 0$ we have that all coefficients in A, b are also larger or equal to 0 (i.e., the positive case), the above-mentioned result of Luby and Nisan [20] gives a fast $\mathsf{polylog}$-space approximation algorithm. In contrast, by our second result, if only $x, c \geq 0$, a small space approximation algorithm is unlikely.

1.3 Organization of the Paper

We prove the first result in Section 2 and the second in Section 3.

2. A CLASSICAL APPROACH

THEOREM 1. *There exists a fixed family of polyhedrons $H = \{H_t\}_{t \in \mathbb{N}}$ such that the following holds: For every language L computable by a Turing Machine with polynomial runtime $t = t(n)$, there exists a* $\mathsf{log\text{-}space}$ *reduction, that converts any instance $x^* \in \{0,1\}^n$ into a linear program with the polyhedron $H_{t(n)}$ (and an objective function that depends on x^*), such that if $x^* \in L$ then the maximum value of the objective function on the polyhedron is at least 1, and if $x^* \notin L$ then the maximum value of the objective function on the polyhedron is at most 0.*

High-level idea.

Our starting point is the results by Dobkin, Lipton and Reiss [8] and Serna [23], that show that any language L, computable by a Turing Machine with polynomial runtime $t = t(n)$, has a $\mathsf{log\text{-}space}$ reduction that converts any instance $x^* \in \{0,1\}^n$ into a linear program of size $\mathsf{poly}(t(n))$, such that if $x^* \in L$ then the the maximum value of the objective function on the polyhedron is 1, and if $x^* \notin L$ then the maximum value of the objective function on the polyhedron is 0. The polyhedron of the resulting linear program is not fixed and *does depend* on x^*.

We show how to convert this linear program into a linear program with almost the same value, and with an a priori *fixed* polyhedron that is *independent* of x^* and L, and depends only on the runtime t (and the length of x^*).

The main idea is to show that there exists a universal set of inequalities such that by "switching off" some of them one

[1]The polyhedron also does not depend on the language L, only on its runtime.

can obtain *any* polyhedron. Then we show how one can use the objective function to effectively "switch off" inequalities.

2.1 Linear Programming is P-Complete

Before we prove Theorem 1, we need to recall the reduction of [8].

Theorem 2 ([8]). *Any language L computable by a Turing Machine with polynomial runtime $t = t(n)$, has a* log-space *reduction that converts any instance $x^* \in \{0,1\}^n$ into a linear program* LP*:*

$$\text{maximize } c \cdot x$$
$$\text{subject to } Ax \geq b,$$

such that the following holds:

1. *All the coefficients in c, b and A are in $\{0, 1, -1\}$.*
2. *If $x^* \in L$ then the value of* LP *is 1, and if $x^* \notin L$ then the value of* LP *is 0.*
3. *The number of variables in* LP *is $d = t^2(n)$, and the number of linear constraints is at most $5d$.*

Proof. Fix any language L computable by a Turing machine with polynomial runtime $t = t(n)$. The log-space reduction that converts any $x^* \in \{0,1\}^n$ into a linear program LP, is defined as follows.

Let C be a Boolean circuit of size $t^2(n)$ that takes as input $x^* \in \{0,1\}^n$ and outputs 1 if and only if $x^* \in L$. (Such a circuit can be built from the Turing machine computing L; see, e.g., [1, Theorem 6.6].) Assume (without loss of generality) that the circuit C has only NOT and AND gates. Denote the number of wires in C by $d = t^2(n)$. The linear program LP consists of d variables, $x_1, \ldots, x_d$, one variable for each wire of C. Denote the n input wires by $x_1, \ldots, x_n$, and denote the output wire by x_d. The objective function of LP is

$$\max x_d.$$

The linear constraints of LP are the following: For every $i \in [d]$, we have the constraint $0 \leq x_i \leq 1$, that can be written as

$$x_i \geq 0$$
$$-x_i \geq -1.$$

For every $i \in [n]$ (corresponding to an input wire), we have the constraint $x_i = x_i^*$, that can be written as

$$x_i \geq x_i^* \tag{1}$$
$$-x_i \geq -x_i^*. \tag{2}$$

For every NOT gate in C, with output wire i and input wire j, we have the constraint $x_i = 1 - x_j$, that can be written as

$$x_i + x_j \geq 1 \tag{3}$$
$$-x_i - x_j \geq -1. \tag{4}$$

For every AND gate with output wire i and input wires j and k, we have the constraints $x_i \leq x_j$, $x_i \leq x_k$, and $x_i \geq x_j + x_k - 1$. These constraints can be written as

$$x_j - x_i \geq 0 \tag{5}$$
$$x_k - x_i \geq 0 \tag{6}$$
$$x_i - x_j - x_k \geq -1. \tag{7}$$

Note that all the coefficients (both in the objective function and in the linear constraints) are in $\{0, 1, -1\}$, as desired. Note that the polyhedron consists of a single element, corresponding to the value of the wires of C on input x^*. Thus, if $x^* \in L$, then $x_d = 1$, and hence the value of LP is 1, and if $x^* \notin L$ then $x_d = 0$, and hence the value of LP is 0, as desired. Note that the number of variables in LP is $d = t^2(n)$ and the number of linear constraints is $\leq 5d$, as desired. □

2.2 Proof of Theorem 1

Proof. Our starting point is the proof of Theorem 2 (see Section 2.1) that for a language L, computable by a Turing Machine with a polynomial runtime $t = t(n)$, gives a log-space reduction that converts any instance $x^* \in \{0,1\}^n$ into a linear program LP:

$$\max c \cdot x$$
$$\text{s.t. } Ax \geq b.$$

We define a universal family of polyhedrons $\{H_d\}$, and show a log-space reduction that converts the linear program LP into a linear program LP′, with the polyhedron H_d. The polyhedron H_d depends on d, but is otherwise independent of x^* and L.

Let

$$T = 2^{4d}. \tag{8}$$

Let $m = 5d$. We will assume for simplicity and without loss of generality that $d \geq 10$. The linear program LP′ consists of $4dm+3m+d$ variables: For every $i \in [d]$, it has the variables x_i corresponding to x_i in the original LP. For every $i \in [d]$ and every $j \in [m]$, it has the variable $y_{j,i}$, where each $y_{j,i}$ supposedly corresponds to the value of $A_{j,i} \cdot x_i$. In addition, for every $i \in [d]$ and every $j \in [m]$, it contains auxiliary variables

$$z_{j,i,0}, z_{j,i,1}, z_{j,i,-1},$$

and for every $j \in [m]$ it contains auxiliary variables

$$w_{j,0}, w_{j,1}, w_{j,-1}.$$

These auxiliary variables are used to "turn off" some of the constraints.

We next define the polyhedron H_d to consist of the following inequalities:[2]

1. For every $i \in [d]$ and $j \in [m]$,

$$y_{j,i} - z_{j,i,0} \leq 0$$
$$y_{j,i} - z_{j,i,1} \leq x_i$$
$$y_{j,i} - z_{j,i,-1} \leq -x_i$$
$$z_{j,i,0} \geq 0$$
$$z_{j,i,1} \geq 0$$
$$z_{j,i,-1} \geq 0$$

[2]We remark that the A part of the linear program LP produced by Theorem 2 is already independent of the input x^*. Using this fact, we could slightly simplify our proof by avoiding Item 1 below. We choose not to use this fact in order to keep the reduction more general, and also in order for the resulting polyhedron to be more natural.

High-level idea.
Intuitively, these six inequalities encode the single inequality $y_{j,i} \leq A_{j,i} \cdot x_i$, in a way that does not depend on $A_{j,i}$. More specifically, we use the objective function to effectively "turn off" the inequalities $y_{j,i} - z_{j,i,\ell} \leq \ell \cdot x_i$, for $\ell \neq A_{j,i}$. This is done by keeping the variables $\{z_{j,i,\ell}\}_{\ell \neq A_{j,i}}$ free, by not including them in the objective function, so that by choosing $z_{j,i,\ell}$ large enough these inequalities can be trivially satisfied without affecting the objective function. On the other hand, we add the term $-T \cdot z_{j,i,A_{j,i}}$ to the objective function, so that even taking $z_{j,i,A_{j,i}}$ exponentially small, would add an exponential large penalty to the objective value. Thus, effectively the remaining inequality $y_{j,i} - z_{j,i,A_{j,i}} \leq A_{j,i} \cdot x_i$ is almost equivalent to the inequality $y_{j,i} \leq A_{j,i} \cdot x_i$.

2. For every $j \in [m]$,

$$\begin{aligned} y_{j,1} + \cdots + y_{j,d} + w_{j,0} &\geq 0 \\ y_{j,1} + \cdots + y_{j,d} + w_{j,1} &\geq 1 \\ y_{j,1} + \cdots + y_{j,d} + w_{j,-1} &\geq -1 \\ w_{j,0} &\geq 0 \\ w_{j,1} &\geq 0 \\ w_{j,-1} &\geq 0 \end{aligned}$$

High-level idea.
Intuitively, these six inequalities encode the single inequality

$$y_{j,1} + \cdots + y_{j,d} \geq b_j,$$

in a way that does not depend on b_j. More specifically, as before, this is done by "turning off" the inequalities $y_{j,1} + \cdots + y_{j,d} + w_{j,\ell} \geq \ell$ for $\ell \neq b_j$, and using the objective function to effectively restrict w_{j,b_j} to be exponentially small, and as a result the equation $y_{j,1} + \cdots + y_{j,d} + w_{j,b_j} \geq b_j$ is almost equivalent to the equation $y_{j,1} + \cdots + y_{j,d} \geq b_j$.

3. For every $i \in [d]$,

$$0 \leq x_i \leq 1$$

Note that the polyhedron H_d is indeed independent of LP, given d.

We next define the linear objective function of LP'. Intuitively, the objective function is the same as that of LP, with additional terms that are used to "turn on" all the inequalities in H_d that correspond to $Ax \geq b$.

We define the linear objective function of LP' to be

$$\max \sum_{i=1}^{d} c_i \cdot x_i - T \cdot \sum_{i \in [d], j \in [m]} z_{j,i,A_{j,i}} - T \cdot \sum_{j \in [m]} w_{j,b_j}. \quad (9)$$

High-level idea.
Intuitively, all the linear constraints that include a variable in $\{z_{j,i,\ell}\}_{\ell \neq A_{j,i}} \cup \{w_{j,\ell}\}_{\ell \neq b_j}$ are "turned off" since we are free to choose these variables to be as large as we want, *without affecting the objective function.* Moreover, as we argue formally below, since we chose T to be large enough (recall that we set $T = 2^{4d}$, see Equation (8)), the maximal value of LP' is obtained when $z_{j,i,A_{j,i}}$ and w_{j,b_j} are exponentially small, and thus the value of LP' is exponentially close to the value of LP.

Formally, we argue that the values of the linear programs LP and LP' are exponentially close, as follows. We define another linear program LP''. We prove that LP'' has the same value as LP', and we prove that the values of LP'' and LP are exponentially close.

The linear program LP'' is defined as follows: The objective function of LP'' is identical to that of LP' (Equation (9)), and the linear constraints consist of a subset of the linear constraints of H_d. Specifically, LP'' only contains the linear constraints in LP' that are "turned on". In other words, LP'' only contains the constraints in LP' that do not contain variables from the set

$$\{z_{j,i,\ell}\}_{\ell \neq A_{j,i}} \cup \{w_{j,\ell}\}_{\ell \neq b_j}.$$

In other words, the linear program LP'' contains the following constraints:

1. For every $i \in [d]$ every $j \in [m]$,

$$\begin{aligned} y_{j,i} - z_{j,i,A_{j,i}} &\leq A_{j,i} \cdot x_i \\ z_{j,i,A_{j,i}} &\geq 0 \end{aligned}$$

2. For every $j \in [m]$,

$$\begin{aligned} y_{j,1} + \cdots + y_{j,d} + w_{j,b_j} &\geq b_j \\ w_{j,b_j} &\geq 0. \end{aligned}$$

3. For every $i \in [d]$,

$$0 \leq x_i \leq 1$$

We next prove that LP' and LP'' have the same value. Clearly the value of LP'' is at least as large as the value of LP', since it has the same objective function and contains only a subset of the constraints. On the other hand, the value of LP'' is not larger than the value of LP', since any solution for LP'' can be extended to a solution in H_d. This is the case since by the definition of LP'', any linear constraint in H_d that does not appear in LP'' contains a variable from

$$\{z_{j,i,\ell}\}_{\ell \neq A_{j,i}} \cup \{w_{j,\ell}\}_{\ell \neq b_j}.$$

Note that these variables do not appear in the objective function, and thus can be set to be arbitrarily large, so as to satisfy the inequalities in LP' that are not in LP'', without affecting the linear objective function.

It remains to argue that the value of LP'' is exponentially close to the value of LP. To this end, we first note that the value of LP'' is at least as large as the value of LP, since by setting all the auxiliary variables $z_{j,i,A_{j,i}}$ and w_{j,b_j} in LP'' to be 0, we obtain a linear program that is equivalent to LP.

We next argue that if the value of LP'' is v then the value of LP is at least $v - 2^{-d}$. To this end, let (x, y, z, w) be a point in the polyhedron of LP'' which yields the maximal value v. We note that it must be the case that all the coordinates of z and all the coordinates of w are of size at most 2^{-3d}. This is the case since otherwise, the linear objective function of LP'' on the point (x, y, z, w) obtains the value

$$\sum_{i=1}^{d} c_i \cdot x_i - T \cdot \sum_{i \in [d], j \in [m]} z_{j,i,A_{j,i}} - T \cdot \sum_{j \in [m]} w_{j,b_j} \leq d - T \cdot 2^{-3d} < 0,$$

which we know is not the maximal value on the polyhedron, since the value of LP'' is at least as large as the value of LP, which is at least 0.

The fact that the coordinates of z and w are exponentially small implies that the point (x, y, z, w) satisfies the constraint

$$Ax \geq b - \varepsilon, \tag{10}$$

for

$$\varepsilon := \sum z_{j,i,A_{j,i}} + \sum w_{j,b_j} \leq 2^{-2d}.$$

We next argue that this implies that there exists x' in the polyhedron of LP (i.e., $Ax' \geq b$), such that for every $i \in [d]$,

$$|x_i - x'_i| \leq 2^{-d}.$$

Recall that $x_1, \ldots, x_d$ correspond to the wires of a circuit C that takes as input $x^* \in \{0,1\}^n$ and outputs 1 if and only if $x^* \in L$ (see the proof of Theorem 2 in Section 2.1). Recall that the n input wires are $x_1, \ldots, x_n$, and the output wire is x_d. Assume without loss of generality that $x_1, \ldots, x_d$ are ordered in an order that agrees with the circuit, that is, for every gate in the circuit, the variable that corresponds to the output of the gate appears after all variables that correspond to the inputs for the gate. Denote by $x'_1, \ldots, x'_d$ the true values of the wires of C on the input $x^* \in \{0,1\}^n$, and note that for every $i \in [n]$ we have $x'_i = x^*_i$, and that the point $x' = (x'_1, \ldots, x'_d)$ is in the polyhedron of LP (it is actually the unique point in the polyhedron of LP).

We next argue that for every $i \in [d]$,

$$|x_i - x'_i| \leq 2^i \cdot \varepsilon.$$

This is proved by induction on i, using Equation (10) and by the definition of LP (see the Proof of Theorem 2 in Section 2.1) as follows:

1. If i corresponds to an input of the circuit C, then by Equation (10), Equation (1) and Equation (2), we have

   $$|x_i - x'_i| \leq \varepsilon\ .$$

2. If i corresponds to the output of a NOT gate with input wire j, then by Equation (10), Equation (3) and Equation (4), and by the inductive hypothesis and the triangle inequality, and since $x'_i = (1 - x'_j)$ we have

   $$|x_i - x'_i| \leq |x_i - (1-x_j)| + |(1-x_j) - (1-x'_j)| + |(1-x'_j) - x'_i|$$

   $$\leq \varepsilon + 2^j \cdot \varepsilon + 0 \leq 2^i \cdot \varepsilon$$

3. If i corresponds to the output of an AND gate with input wires j, k (for simplicity and without loss of generality, we assume $j \neq k$), then by Equation (10), Equation (5), Equation (6), Equation (7) and by the inductive hypothesis and since $0 \leq x_i, x_j, x_k \leq 1$ and $j, k < i$, we have

   (a) If $x'_i = 0$ then either $x'_j = 0$ or $x'_k = 0$. Without loss of generality assume $x'_j = 0$. Thus,

   $$x_i - x'_i = x_i \leq x_j + \varepsilon \leq x'_j + 2^j \cdot \varepsilon + \varepsilon \leq 0 + 2^i \cdot \varepsilon$$

   and

   $$x_i - x'_i = x_i \geq 0$$

   (b) If $x'_i = 1$ then $x'_j = x'_k = 1$. Thus,

   $$x_i - x'_i = x_i - 1 \leq 0$$

   and

   $$\begin{aligned} x_i - x'_i &\geq (x_j + x_k - 1 - \varepsilon) - 1 \\ &= (x_j - x'_j) + (x_k - x'_k) - \varepsilon \\ &\geq -2^j \cdot \varepsilon - 2^k \cdot \varepsilon - \varepsilon \geq -2^i \cdot \varepsilon \end{aligned}$$

In particular, we proved that $|x_d - x'_d| \leq 2^d \cdot \varepsilon \leq 2^{-d}$. Since the value of LP'' is at most x_d and the value of LP is x'_d, we proved that if the value of LP'' is v then the value of LP is at least $v - 2^{-d}$.

Thus LP'', and hence also LP' satisfies that if $x^* \in L$ then the maximum value of the objective function on the polyhedron is at least $1 - 2^{-d}$, and if $x^* \notin L$ then the maximum value of the objective function on the polyhedron is at most 2^{-d}. The gap between $1 - 2^{-d}$ and 2^{-d} can be easily amplified to a gap between 1 and 0, by applying a linear function on the objective function, as described in the introduction. $\square$

3. A PCP-BASED APPROACH

In this section, we consider the problem of *approximating* Linear Programming with a fixed polyhedron. For us to be able to meaningfully talk about approximation, we restrict to the case of semi-positive linear programs, i.e, when $c, x \geq 0$. While the hardness result of Section 2 can be adapted to this case, the resulting approximation factors are miniscule. In this section we show hardness for approximation factors $2^{\mathsf{polylog}(n)}$ through a $\mathsf{polylog}$ space and $\mathsf{quasi\text{-}poly}$ time reduction, thus showing that $2^{(\log n)^{o(1)}}$-space approximation algorithms are unlikely.

Theorem 3. *There exists a fixed family of polyhedrons $H = \{H_t\}_{t \in \mathbb{N}}$ such that the following holds: For every language $L \in \mathsf{P}$ computable by a Turing Machine with polynomial runtime $t = t(n)$, there exists a* $\mathsf{polylog}$ *space and* $\mathsf{quasi\text{-}poly}$ *time reduction, that converts any instance $x^* \in \{0,1\}^n$ into a linear program with the polyhedron $H_{t(n)}$, and an objective function* $\max c \cdot x$, *such that $x, c \geq 0$, and such that if $x \in L$ then the maximum value of the objective function on the polyhedron is 1, and if $x \notin L$ then the maximum value of the objective function on the polyhedron is smaller than $2^{-\mathsf{polylog}(n)}$.*

The proof of Theorem 3 uses a recent result of Kalai, Raz and Rothblum (stated in Theorem 5 below), showing that every language in P has a no-signaling multi-prover interactive proof with poly-logarithmic communication complexity.

In Section 3.1 below, we provide the necessary background for proving Theorem 3, and then in Section 3.2 we prove Theorem 3.

3.1 Preliminaries

3.1.1 Notation

For a vector $a = (a_1, \ldots, a_k)$ and a subset $S \subseteq [k]$, we denote by a_S the sequence of elements of a that are indexed by indices in S, that is, $a_S = (a_i)_{i \in S}$. In general, we denote by a_S a sequence of elements indexed by S, and we denote by a_i the i^{th} coordinate of a vector a.

For a distribution $\mathcal{A}$, we denote by $a \in_R \mathcal{A}$ a random variable distributed according to $\mathcal{A}$ (independently of all other random variables).

3.1.2 *Multi-Prover Interactive Proofs*

Multi-prover interactive proofs (MIPs) were introduced by [5]. In such a proof system a set of provers wish to convince a verifier of the validity of a statement. Specifically, let L be a language and let x be an input of length n. In a one-round k-prover interactive proof, k computationally unbounded provers, $P_1, \ldots, P_k$, try to convince a (probabilistic) $\mathsf{poly}(n)$-time verifier, V, that $x \in L$. The input x is known to all parties.

The proof consists of only one round. Given x and her random string, the verifier generates k queries, $q_1, \ldots, q_k$, one for each prover, and sends them to the k provers. Each prover responds with an answer that depends only on her own individual query. That is, the provers respond with answers $a_1, \ldots, a_k$, where for every i we have $a_i = P_i(q_i)$. Finally, the verifier decides whether to accept or reject based on the answers that she receives (as well as the input x and her random string).

We say that $(V, P_1, \ldots, P_k)$ is a one-round multi-prover interactive proof system (MIP) for L if the following two properties are satisfied:

1. **Completeness:** For every $x \in L$, the verifier V accepts with probability 1, after interacting with $P_1, \ldots, P_k$.
2. **Soundness:** For every $x \notin L$, and any (computationally unbounded, possibly cheating) provers $P_1^*, \ldots, P_k^*$, the verifier V rejects with probability $\geq 1 - \varepsilon$, after interacting with $P_1^*, \ldots, P_k^*$, where ε is a parameter referred to as the *error* or *soundness* of the proof system.

Other important parameters of an MIP include the number of provers, the length of queries, the length of answers, and the error. One celebrated result in this line of work is that of Babai, Fortnow, and Lund [3] who showed that any language in NEXP has an MIP with negligible soundness.

3.1.3 *No-Signaling* MIPs

We consider a variant of the MIP model, where the cheating provers are more powerful. In the MIP model, each prover answers her own query locally, without knowing the queries that were sent to the other provers. The no-signaling model allows each answer to depend on all the queries, as long as for any subset $S \subset [k]$, and any queries q_S for the provers in S, the distribution of the answers a_S, conditioned on the queries q_S, is independent of all the other queries.

Intuitively, this means that the answers a_S do not give the provers in S information about the queries of the provers outside S, except for information that they already have by seeing the queries q_S.

Formally, denote by D the alphabet of the queries and denote by Σ the alphabet of the answers. For every $q = (q_1, \ldots, q_k) \in D^k$, let $\mathcal{A}_q$ be a distribution over Σ^k. We think of $\mathcal{A}_q$ as the distribution of the answers for queries q.

We say that the family of distributions $\{\mathcal{A}_q\}_{q \in D^k}$ is *no-signaling* if for every subset $S \subset [k]$ and every two sequences of queries $q, q' \in D^k$, such that $q_S = q'_S$, the following two random variables are identically distributed:

- a_S, where $a \in_R \mathcal{A}_q$, and
- a'_S, where $a' \in_R \mathcal{A}_{q'}$.

An MIP $(V, P_1, \ldots, P_k)$ for a language L is said to have soundness ε against no-signaling strategies (or provers) if the following (more general) soundness property is satisfied:

2. **Soundness:** For every $x \notin L$, and any no-signaling family of distributions $\{\mathcal{A}_q\}_{q \in D^k}$, the verifier V rejects with probability $\geq 1 - \varepsilon$, where on queries $q = (q_1, \ldots, q_k)$ the answers are given by $(a_1, \ldots, a_k) \in_R \mathcal{A}_q$, and ε is the error parameter.

The study of multi-prover interactive proofs (MIPs) that are secure against no-signaling provers was motivated by the study of MIPs with provers that share entangled quantum states. No-signaling provers are more powerful than entangled provers, since no-signaling provers are allowed to use arbitrary strategies, as long as their strategies cannot be used for communication between any two disjoint sets of provers. By the physical principle that information cannot travel faster than light, a consequence of Einstein's special relativity theory, it follows that all the strategies that can be realized by provers that share entangled quantum states are no-signaling strategies.

No-signaling strategies were first studied in physics in the context of Bell inequalities by Khalfin and Tsirelson [19] and Rastall [22], and they gained much attention after they were reintroduced by Popescu and Rohrlich [21]. MIPs that are secure against no-signaling provers were extensively studied in the literature (see for example [24, 4, 2, 17, 13, 11, 12]). It was known that they are contained in EXP, and recently [14] showed that they also contain EXP, thus giving a full characterization of their exact power.

Informal Theorem 4 ([14]). *For any language L computable in time $t = t(n)$, there exists an* MIP *with soundness error $2^{-\mathsf{polylog}(t)}$ against no-signaling cheating provers. The number of provers and the communication complexity is $\mathsf{polylog}(t)$. The verifier runs in time $n \cdot \mathsf{polylog}(t)$ (and the provers run in time $\mathsf{poly}(t)$). Moreover, the verifier only runs in time $\mathsf{polylog}(t)$ if he is given oracle access to a (specific) encoding of x,*[3] *where each entry of the encoding can be computed deterministically from x in time $\tilde{O}(n)$ and space $O(\log n)$.*

In this work, we use this theorem for languages in P. Note that this theorem implies that for languages in P the verifier runs in $\tilde{O}(n)$ time and in $\mathsf{polylog}(n)$ space. Thus, we restate the theorem as follows.

Theorem 5 ([14]). *If $L \in \mathsf{P}$, then there exists an* MIP *for L with $\mathsf{polylog}(n)$ provers, and with soundness error $2^{-\mathsf{polylog}(n)}$ against no-signaling strategies. The verifier runs in time $\tilde{O}(n)$, space $\mathsf{polylog}(n)$, and tosses at most $\mathsf{polylog}(n)$ coins (and the provers run in polynomial time). Each query and answer is of length $\mathsf{polylog}(n)$.*

We use Theorem 5 to show a reduction from any language $L \in \mathsf{P}$ to a linear program. Our reduction runs in quasi-poly time and polylog space. In particular, our reduction takes an instance of size n and converts it into a linear program of size quasi-polynomial in n, where the polyhedron is on

[3]This encoding is the low-degree extension encoding. We refer the reader to [14] for details.

quasi-polynomial number of variables (i.e., quasi-polynomial dimensions). This polyhedron is fixed, independent of the instance x (and depends only on its size $n = |x|$).[4]

3.2 Proof of Theorem 3

PROOF. Let L be any language in P. By Theorem 5, the language L has an MIP, $(V, P_1, \ldots, P_k)$, where $k = \mathsf{polylog}(n)$, with communication complexity $\mathsf{polylog}(n)$ and soundness $2^{-\mathsf{polylog}(n)}$ against no-signaling provers (where n is the instance size).

We define a reduction $\mathcal{R}$ that takes as input an instance $x \in \{0,1\}^n$ and converts it into a linear program, as follows: Consider all possible no-signaling families of distributions of cheating provers in the MIP. For each such possible no-signaling family of distributions $\{\mathcal{A}_q\}_{q \in D^k}$, denote by

$$p_{q,a} = \Pr_{A \in_R \mathcal{A}_q}[A = a].$$

Note that $\{\mathcal{A}_q\}_{q \in D^k}$ is a no-signaling family of distributions if and only if the following conditions are satisfied (the first two conditions hold if and only if each $\mathcal{A}_q$ is a distribution, and the last condition holds if and only if these distributions are no-signaling):

1. For every $q = (q_1, \ldots, q_k) \in D^k$ and for every $a \in \Sigma^k$,
$$p_{q,a} \geq 0.$$
2. For every $q = (q_1, \ldots, q_k) \in D^k$,
$$\sum_{a \in \Sigma^k} p_{q,a} = 1.$$
3. For every $S \subseteq [k]$, for every $q = (q_1, \ldots, q_k) \in D^k$ and $q' = (q'_1, \ldots, q'_k) \in D^k$ for which $q_S = q'_S$, and for every $a_S \in \Sigma^S$, it holds that
$$\sum_{a' : a'_S = a_S} p_{q,a'} = \sum_{a' : a'_S = a_S} p_{q',a'}.$$

Denote by p_q the probability that V sends the provers queries $q = (q_1, \ldots, q_k) \in \mathcal{D}^k$. The fact that $(V, P_1, \ldots, P_k)$ is an MIP that is secure against no-signaling strategies (with soundness $2^{-\mathsf{polylog}(n)}$ and perfect completeness), implies that if $x \notin L$ then[5]

$$\sum_q p_q \sum_{a : V(x,q,a)=1} p_{q,a} \leq 2^{-\mathsf{polylog}(n)},$$

and if $x \in L$ then there exists a (classical) strategy for which

$$\sum_q p_q \sum_{a : V(x,q,a)=1} p_{q,a} = 1.$$

[4]The polyhedron is also independent of the language L, and depends only on its time complexity.

[5]Here we assume for simplicity that the verifier's decision is a function only of the input x, the queries q, and the answers a, and not of the verifier's randomness. This is indeed the case for the verifier given by Theorem 5 (and in fact can always be assumed without loss of generality).

Thus, the reduction $\mathcal{R}$ converts $x \in \{0,1\}^n$ into the linear program with the polyhedron defined by:

$$p_{q,a} \geq 0, \ \forall q \in D^k \text{ and } \forall a \in \Sigma^k.$$
$$\sum_{a \in \Sigma^k} p_{q,a} = 1, \ \forall q \in D^k.$$
$$\sum_{a' : a'_S = a_S} p_{q,a'} = \sum_{a' : a'_S = a_S} p_{q',a'},$$
$$\forall S \subseteq [k], \ \forall q, q' \in D^k \text{ s.t. } q_S = q'_S, \ \forall a_S \in \Sigma^S.$$

Note that this polyhedron is fixed and does not depend on the instance x. The objective function is

$$\max_{\{p_{q,a}\}} \sum_q p_q \sum_{a : V(x,q,a)=1} p_{q,a}, \tag{11}$$

where for every q, p_q is a fixed value defined by the verifier in the underlying MIP, and $\{p_{q,a}\}$ are the variables. Note that if $x \in L$ then the maximum of this objective function on the polyhedron is 1, whereas if $x \notin L$ then the maximum of this objective function on the polyhedron is at most $2^{-\mathsf{polylog}(n)}$. Thus, determining whether x is in the language or not reduces to approximating the objective function.

It remains to prove that the space complexity of $\mathcal{R}$ is $\mathsf{polylog}(n)$ (and hence the runtime is at most $\mathsf{quasi\text{-}poly}(n)$). Since the polyhedron is fixed, it suffices for the reduction $\mathcal{R}$ to generate the objective function, as defined in Equation (11).[6] Namely, $\mathcal{R}$ needs to compute p_q for every q, and $V(x, q, a)$ for every q and a. $\mathcal{R}$ computes p_q by enumerating over all possible random coin tosses of the MIP verifier (recall that the MIP verifier tosses at most $\mathsf{polylog}(n)$ coins). This, together with the fact that the space complexity of V is $\mathsf{polylog}(n)$, implies that the space complexity of $\mathcal{R}$ is $\mathsf{polylog}(n)$, as desired. $\square$

Acknowledgments.
We thank Boaz Barak for illuminating discussions.

4. REFERENCES

[1] S. Arora and B. Barak. *Computational complexity: A modern approach.* Cambridge University Press, Cambridge, 2009.

[2] D. Avis, H. Imai, and T. Ito. On the relationship between convex bodies related to correlation experiments with dichotomic observables. *Journal of Physics A: Mathematical and General, 39(36)*, 39(36):11283, 2006.

[3] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 16–25. IEEE Computer Society, 1990.

[4] J. Barrett, N. Linden, S. Massar, S. Pironio, S. Popescu, and D. Roberts. Nonlocal correlations as an information-theoretic resource. *Physical Review A, 71(022101)*, 71(2):022101, 2005.

[5] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to

[6]We remark that the inequalities describing the polyhedron can easily be computed in $\mathsf{polylog}(n)$ space.

remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 113–131, 1988.

[6] D. Bertsimas and S. Vempala. Solving convex programs by random walks. *J. ACM*, 51(4):540–556, 2004.

[7] G. B. Dantzig. Maximization of linear function of variables subject to linear inequalities. pages 339–347, 1951.

[8] D. P. Dobkin, R. J. Lipton, and S. P. Reiss. Linear programming is log-space hard for P. *Inf. Process. Lett.*, 8(2):96–97, 1979.

[9] J. Dunagan and S. Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 315–320, 2004.

[10] U. Feige and J. Kilian. Making games short (extended abstract). In F. T. Leighton and P. W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 506–516. ACM, 1997.

[11] T. Holenstein. Parallel repetition: Simplification and the no-signaling case. *Theory of Computing*, 5(1):141–172, 2009.

[12] T. Ito. Polynomial-space approximation of no-signaling provers. In *ICALP (1)*, pages 140–151, 2010.

[13] T. Ito, H. Kobayashi, and K. Matsumoto. Oracularization and two-prover one-round interactive proofs against nonlocal strategies. In *IEEE Conference on Computational Complexity*, pages 217–228, 2009.

[14] Y. T. Kalai, R. Raz, and R. D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 485–494, 2014.

[15] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–396, 1984.

[16] J. A. Kelner and D. A. Spielman. A randomized polynomial-time simplex algorithm for linear programming. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 51–60, 2006.

[17] J. Kempe, H. Kobayashi, K. Matsumoto, B. Toner, and T. Vidick. Entangled games are hard to approximate. In *FOCS*, pages 447–456, 2008.

[18] L. G. Khachiyan. A polynomial algorithm in linear programming. *In Doklady Akademia Nauk SSSR*, pages 1093–1096, 1979.

[19] L. A. Khalfin and B. S. Tsirelson. Quantum and quasi-classical analogs of Bell inequalities. In *In Symposium on the Foundations of Modern Physics*, pages 441–460, 1985.

[20] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In S. R. Kosaraju, D. S. Johnson, and A. Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 448–457. ACM, 1993.

[21] S. Popescu and D. Rohrlich. Quantum nonlocality as an axiom. *Foundations of Physics*, 24(3):379–385, 1994.

[22] P. Rastall. Locality, Bell's theorem, and quantum mechanics. *Foundations of Physics*, 15(9):963–972, 1985.

[23] M. J. Serna. Approximating linear programming is log-space complete for P. *Inf. Process. Lett.*, 37(4):233–236, 1991.

[24] B. Toner. Monogamy of non-local quantum correlations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 465(2101):59–69, 2009.