Appears in *Proceedings of the Second Israel Symposium on Theory and Computing Systems*, IEEE, June 1993. Preliminary version in IBM Research Report RC 17969, May 1992.

Interactive Proofs and Approximation: Reductions from Two Provers in One Round

Mihir Bellare*

Abstract

We present hard to approximate problems in the following areas: systems of representatives, network flow, and longest paths in graphs. In each case we show that there exists some $\delta > 0$ such that polynomial time approximation to within a factor of $2^{\log^{\delta} n}$ of the optimal implies that NP has quasi polynomial time deterministic simulations. The results are derived by reduction from two prover, one round proof systems, and exemplify the ability of such reductions to yield hardness of approximations results for many different kinds of problems.

*Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093. E-mail: mihir@cs.ucsd.edu

1 Introduction

The connection between (multi-prover) interactive proofs and approximation which originates in [FGLSS] has inaugurated a new and powerful approach to resolving the approximation complexity of optimization problems. Typically, one obtains results of the following form:

> The existence of a polynomial time approximation algorithm for problem P implies that NP has efficient deterministic simulations.

What makes these results strong is that they hold even for approximation algorithms which return approximations of very poor quality; on the other hand the efficiency in the deterministic solution of NP is low—polynomial or quasi-polynomial. For example, the existence of a $n^{\Theta(1)}$ factor approximation algorithm for MAX CLIQUE implies P = NP [FGLSS, AS, ALMSS].

These results are established by reductions. Two kinds of reductions have been used. First, there are reductions made directly from interactive proofs. Second, there are reductions from optimization problems already shown hard to approximate by the first method. The focus of this paper is on the first kinds of reductions.

1.1 Direct Reductions

A direct reduction (from multi-prover interactive proofs), namely the MAX CLIQUE reduction [FGLSS], is of course where this area began, so such reductions are certainly important. Perhaps what is most interesting, however, is that they have remained important in proving new problems are hard to approximate. Let us begin, however, by saying in more detail what these reductions are. The discussions here will be informal, and we refer the reader to Section 2 for definitions. For simplicity we focus on maximization problems.

What is a direct reduction? Let V be a verifier for a multi-prover interactive proof of membership in an NP-complete language L, and let P be an optimization problem which we wish to show is hard to approximate. In its simplest form, a direct reduction Φ associates to a string x an instance $\Phi(x)$ of P such that the optimal solution to $\Phi(x)$ has a value which is the same, up to a multiplicative factor, as the probability that the verifier accepts in conversation with the optimal provers.

The "gap" in the accepting probability—it is 1 if $x \in L$ and at most some small error probability ϵ otherwise—is translated into a gap in the values of optimal solutions to $\Phi(x)$, and thus approximating the value of an optimal solution to an instance of problem P yields a decision procedure for L.[1]

The importance of complexity: 2P1R reductions. The MAX CLIQUE reduction was done from an arbitrary (multi-prover) proof system. An important element in finding direct reductions for new problems has been the use of proof systems which minimize the number of provers. Reductions from two prover, one round (2P1R) proofs were used to show that QUADRATIC PROGRAMMING is hard to approximate [BR, FL] and are used also in this paper; subsequently they have been used in many places (cf. Section 1.3). Indeed these proof systems are very simple, and it seems natural that one can reduce them to a larger set of problems.

[1] Once a reduction is known, the strength of the non-approximability result that follows—as measured by the quality of approximation proved hard and the efficiency of the deterministic simulation of NP in the conclusion—depends on certain parameters of the proof system, such as the randomness, answer sizes, and error probability. Constructing more efficient proofs leads to stronger results [FGLSS, AS, ALMSS, BGLR]. In this paper we will focus on reductions, the necessary first step to enlarging the class of problems we can show hard to approximate.

The number of provers is important also in the MAX-3SAT reduction of [ALMSS]; they need a constant number of provers, each of whom returns a constant number of bits.[2]

A CONTRAST WITH NP-HARDNESS. It is interesting to contrast the development of hardness of approximation results with the development of hardness of decision results as captured by the theory of NP-hardness. Following Cook's theorem, reductions from known NP-complete problems immediately became the main tool for deriving hardness of decision results. In the approximation domain, however, the use of direct reductions has continued to be useful. It is possible that at some point we may have a large enough "core" of hard to approximate problems that others may be derived merely by reduction from them, but it does seems that direct reductions are important in the early stages.

1.2 Problems and Results

In this paper we use 2P1R reductions to find a collection of hard to approximate problems. We have tried to choose a diverse collection, spanning different traditional "classes" of NP-complete problems. In particular, we have problems about systems of representatives, network flow, and longest paths in graphs. The descriptions of the problems follows. Each is a maximization problem; i.e. maximize the value over the solution space. For explanations and more information on the problems see Section 3; for definitions of optimization problems and notation see Section 2.

MAX CAPACITY REPS
Instance: Disjoint sets $S_1, \ldots, S_m$. For each $i \neq j$ and each $x \in S_i$, $y \in S_j$, a capacity $c(x, y) \geq 0$.
Solutions: An m-set T is a solution if it contains exactly one element of S_i for each $i = 1, \ldots, m$. Such a solution is called a system of representatives of $S_1, \ldots, S_m$.
Value of Solutions: The value of a system of representatives T is $\sum_{x,y \in T} c(x, y)$.

MAX PRIORITY FLOW
Instance: Digraph $G = (V, E)$, sources $s_1, \ldots, s_k \in V$, sinks $t_1, \ldots, t_l \in V$. For each arc $e \in E$ a capacity $c(e) \geq 0$. For each $v \in V$ a bound $b(v) \geq 0$ and a partial order $\prec_v$ on the set of arcs leaving v. We refer to $\prec_v$ as the priority ordering at v.
Solutions: A flow $f\colon E \to \mathsf{R}_+$ is a solution if

- For all $e \in E$: $f(e) \leq c(e)$.
- For all $v \in V - \{s_1, \ldots, s_k, t_1, \ldots, t_l\}$: flow is conserved at v.
- For all $v \in V$: the flow leaving v is at most $b(v)$.
- For all $v \in V$ and all arcs e_1, e_2 leaving v: if $f(e_1) < c(e_1)$ and $e_1 \prec_v e_2$ then $f(e_2) = 0$.

A solution is called a priority flow.
Value of Solutions: The value of a priority flow f is the amount of flow entering sink t_1.

LONGEST CR PATH
Instance: Digraph $G = (V, E)$. For each edge e a weight $w(e) \geq 0$ and a color (red or blue). For each vertex v a label which says either "monochromatic" or "dichromatic."

[2] We speak in terms of provers—rather than queries in a probabilistically checkable proof—for consistency with the rest of the discussions in this paper. In all known reductions, the difference between the models (multi prover proofs as opposed to probabilistically checkable proofs) is unimportant under a rough correspondence of the number of provers with the number of queries. For more information on the relation between the models in the approximation context see [BGLR].

Solutions: A simple path in G is a solution if at each non-endpoint node of the path, the entering and exiting edges are of the same color if the node is monochromatic, and are of different colors if the node is dichromatic. A solution is called a color respecting (CR) path.
Value of Solutions: The value of a CR path is its weight (i.e. the sum of the weights of all its edges).

Note that the standard LONGEST PATH problem is the special case of the LONGEST CR PATH problem in which all edges of the given graph are of weight 1 and identical color, and all nodes are monochromatic.

The main result is that 2P1R reduces to each of the above problems (cf. Propositions 3.1, 3.3 and 3.4). A consequence of this fact is that these problems are hard to approximate in the following sense.

Theorem 1.1 There exist constants $c, \mu > 0$ such that for each of the problems MAX CAPACITY REPS, MAX PRIORITY FLOW and LONGEST CR PATH the following is true.

(1) The existence of a polynomial time, factor $2^{\Theta(\log^{1/c} n)}$ approximation algorithm implies NP $\subseteq \bigcup_{d>0} \mathrm{DTIME}(n^{d \log^c n})$.

(2) The existence of a polynomial time, factor μ approximation algorithm implies P = NP.

Values for the constants c and μ can be found in Remark 2.18. An additional consequence of the fact that 2P1R reduces to each of the above problems is that (natural decision versions of) these problems are NP-complete (cf. Remark 2.20).

1.3 History and Related Work

An early version of this paper appeared as [Be]. Since then there has been much progress in interactive proofs and approximation.

Efficient probabilistically checkable proofs were constructed by [AS, ALMSS], improving [BFL, BFLS, FGLSS]; these results are exploited here (specifically for the second conclusion in Theorem 1.1).

New reductions have established the hardness of approximation of many well known optimization problems such as the minimum chromatic number and the minimum set cover [LY1]. A general result on the hardness of approximation for hereditary graph properties appears in [LY2]. The set cover reduction of [LY1], as well as some of the reductions of [LY2], use 2P1R proofs.

Recently it was shown that the longest path problem (of which our LONGEST CR PATH problem is a variant) is hard to approximate in the sense of Theorem 1.1 [KMR].

Efficient four prover proofs were constructed in [BGLR], leading to improvement of many approximation results including those for set cover, clique, chromatic number, and MAX-3SAT.

2 Proofs, Optimization and Approximation

It is convenient to view a 2P1R proof as an optimization problem; the connection between proofs and optimization is then perhaps most apparent, and one can define reductions like for optimization problems (cf. Section 2.3). We begin by summarizing definitions and notation for 2P1R proofs, optimization problems and approximation. We denote by R_+ the non-negative reals and by R_+^* the positive reals.

2.1 2P1R Proofs

The parties in a two-prover, one-round (2P1R) interactive proof system are a probabilistic, polynomial time verifier V, and a pair of (computationally unbounded, deterministic) provers A and B. We will measure the complexity by a single parameter l. Formally:

Definition 2.1 A verifier of complexity l: $\mathsf{N} \to \mathsf{N}$ is a pair of functions (π, ρ), each computable in time polynomial in the length of its first argument. On input $x \in \{0,1\}^*$ and $R \in \{0,1\}^{l(|x|)}$ the function π returns a pair of strings p, q. On input x and strings p, a, q, b the function ρ returns a bit, and we ask that $\rho(x, p, a, q, b) = 0$ if either $|a|$ or $|b|$ is different from $l(|x|)$. A prover is a map from $\{0,1\}^*$ to $\{0,1\}^*$.

On common input x the parties engage in a simple interaction which is begun by the verifier. The latter applies π to x and his random tape $R \in \{0,1\}^{l(|x|)}$ to get a pair of "questions" p, q. He sends p to A and q to B. A answers with $a = A(p)$, and B with $b = B(q)$. If $\rho(x, p, a, q, b) = 1$ then the verifier is considered to "accept" else to "reject." The provers' (joint) goal is to maximize the probability that V accepts. The complexity l is defined to bound the number of coin tosses and the size of answers sufficient to convince the verifier to accept, because these are the quantities important in applications to approximation (cf. Remark 2.6). In what follows we let $\pi^i(x, R)$ denote the question to the i-th prover, $i = 1, 2$.

Definition 2.2 Let $V = (\pi, \rho)$ be a verifier of complexity l. For each x and each pair (A, B) of provers, let $\mathtt{ACC}_{V,(A,B)}(x)$ denote the probability that

$$\rho(x, \pi^1(x, R), A(\pi^1(x, R)), \pi^2(x, R), B(\pi^2(x, R))) = 1$$

when R is chosen at random from $\{0,1\}^{l(|x|)}$. The accepting probability of the verifier V at x, denoted $\mathtt{ACC}_V(x)$, is the maximum of $\mathtt{ACC}_{V,(A,B)}(x)$ over all possible pairs (A, B) of provers.

Definition 2.3 Suppose L is a language and ϵ a function of N to $[0, 1]$. We say that V has error probability ϵ with respect to L if the following two conditions hold: First, $x \in L$ implies $\mathtt{ACC}_V(x) = 1$; Second, $x \notin L$ implies $\mathtt{ACC}_V(x) < \epsilon(|x|)$.

We say that a language L has a 2P1R proof with complexity l and error probability ϵ if there exists a verifier having complexity l and error probability ϵ with respect to L. Usually we say that L has a 2P1R proof if it has a 2P1R proof with complexity $\mathrm{poly}(n)$ and error probability $1/2$. Important to our results is the fact that languages in NP have 2P1R proofs of very low complexity.

Theorem 2.4 There is a constant $c > 0$ and a constant $\epsilon < 1$ such that the following are true for any language $L \in \mathrm{NP}$.

(1) L has a 2P1R proof with complexity $O(\log^c n)$ and error probability $1/n$.
(2) L has a 2P1R proof with complexity $O(\log n)$ and error probability ϵ.

The first result is due to [FL, LS], improving [Ki, Fe]. The second result is derived by applying a standard transformation (cf. [FRS]) to the main result of [ALMSS].

Remark 2.5 The value of the constant c achieved by [FL, LS] is $c = 3$. For any constant $\epsilon_* > 0$ one can achieve $\epsilon = \frac{1}{2} + \epsilon_*$ by first applying the [FRS] transformation to the [ALMSS] result and then applying [Fe].[3]

[3] In the version of this paper which appears in the ISTCS proceedings it is mistakenly stated that the best known constant is $\epsilon = 71/72$. I am grateful to Feige for the correction.

Remark 2.6 In Definition 2.2 we do not restrict the length of the verifier's questions to $l(|x|)$, and such a restriction is indeed not necessary: in all known reductions (including ours) all that one needs is that the number of possible questions to each prover is bounded by $2^{l(|x|)}$, and this is true if the randomness is $l(|x|)$.[4] This becomes important if the complexity of 2P1R proofs with given error ϵ for a NP complete language L could be reduced at the cost of increasing question sizes, and it is likely that this latitude is useful if one is using randomness reduction techniques to reduce the number of coin tosses.

Remark 2.7 A problem that remains open is to show that every $L \in$ NP has 2P1R proofs with logarithmic complexity and error $1/n$; this result would improve 2P1R based hardness of approximation results such as those of this paper (cf. Remark 2.19) and those of [BR, FL, LY1, LY2]. An even better (and seemingly ideal) result would be that for every function $k(n)$, every $L \in$ NP has 2P1R proofs with logarithmic randomness, answer sizes $O(k(n))$, and error $2^{-k(n)}$; this would lead to improvement of *all* known (interactive proof based) hardness of approximation results.

2.2 Optimization and Approximation

A *(combinatorial) optimization problem* is a pair (S, G); S is a map assigning to each *instance* x a set $S(x)$ called the *solution space*, and $G(x, y)$ is, for each solution $y \in S(x)$, a non-negative real number called the *value* of y with respect to x. An instance x is *degenerate* if $S(x) = \emptyset$. When an instance is degenerate the optimization problem itself is not well defined, and hence we will restrict our attention to non-degenerate instances. For simplicity we focus on maximization problems, so that the problem is to maximize the value over the solution space. If x is non-degenerate then we let $G^*(x) = \sup_{y \in S(x)} G(x, y)$.

Definition 2.8 Let (S, G) be an optimization problem and μ a map from N to R_+^*. A *factor* μ *approximation* for (S, G) is a function $\tilde{G}$ which, on any non-degenerate instance x gives a number $\tilde{G}(x) \in \mathsf{R}_+$ for which

$$\mu(|x|)^{-1} \cdot G^*(x) \leq \tilde{G}(x) \leq G^*(x) \ .$$

Remark 2.9 This definition is for the "decision" rather than the "search" version of the problem, in a strong sense: not only is it not required that an approximation algorithm output a solution (with value $\tilde{G}(x)$), but it is not even required that there exist a solution of value $\tilde{G}(x)$. Since our results are negative, this serves to strengthen them.

Remark 2.10 The use of this particular definition of approximation is not crucial to our results, and has been chosen mainly for simplicity. In fact interactive proof based reductions (and those of this paper in particular) tend to establish strong (negative) results in terms of any of the commonly used definitions.

Remark 2.11 For simplicity we are measuring the quality of the approximation as a function of the *length* of the input. Depending on the problem, one might wish to measure it in terms of some other aspect of the input; for example, for graph problem, it is customary to measure the approximation as a function of the number of nodes in the graph. Our results extend to other such commonly used "norms."

[4] Of course, the question lengths are always bounded by a polynomial in $n = |x|$ (since we assume they are computable in time polynomial in n) and this is important to the reductions.

2.3 The Multi-Prover Optimization Problem

A (2P1R) proof system may be viewed in a natural way as an optimization problem. Given a verifier V the problem on input x is to maximize the probability that V accepts, over the solution space consisting of all pairs of provers. It is convenient to "scale" the probability up so that it is an integer. Formally:

Definition 2.12 Let V be a verifier of complexity l. Its associated optimization problem, $\mathrm{OPT}_V = (S_V, G_V)$, is defined as follows. The solution space $S_V(x)$ associated to any instance x is the set of all pairs of provers (A, B). The value is defined by $G_V(x, (A, B)) = 2^{l(|x|)} \cdot \mathtt{ACC}_{V,(A,B)}(x)$.

By definition $G^*_V(x) = 2^{l(|x|)} \cdot \mathtt{ACC}_V(x)$.

Fix a verifier V having a error probability $\epsilon(\cdot)$ with respect to an NP-complete language L. The feature of the optimization problem OPT_V that is interesting is that the value of an optimal solution is large ($2^{l(|x|)}$) when $x \in L$ and small (at most $\epsilon(|x|) \cdot 2^{l(|x|)}$) otherwise, so that approximating the value of an optimal solution (to within a factor of $\epsilon(\cdot)^{-1}$) corresponds to deciding whether or not $x \in L$. Specifically, OPT_V is "hard to approximate" in the following sense: if there exists a polynomial time, factor $\epsilon(\cdot)^{-1}$ approximation algorithm for OPT_V then P = NP. This observation underlies the interactive proof based paradigm for hardness of approximation results that originates in [Co, FGLSS]. To exploit it to show hardness of approximation of a given optimization problem (S, G), we consider "value preserving" reductions of OPT_V to (S, G). Following the example of the clique reduction of [FGLSS], the reduction is computable in time (polynomial in $|x|$ and) *exponential* in the complexity l of the given verifier, and in particular constructs instances of (S, G) of size (polynomial in $|x|$ and) exponential in l.

Definition 2.13 Let V be a verifier of complexity l and let (S, G) be an optimization problem. Let Φ be a map which associates to an instance x of $\mathrm{OPT}_V = (S_V, G_V)$ a non-degenerate instance $\Phi(x)$ of (S, G), and is computable in time polynomial in $|x| + 2^{l(|x|)}$. We say that Φ is a value preserving reduction of OPT_V to (S, G) if there is a polynomial time computable map $K\colon \{0,1\}^* \to \mathsf{R}^*_+$ such that

$$G^*(\Phi(x)) \;=\; K(x) \cdot G^*_V(x)$$

for all $x \in \{0,1\}^*$.

Equivalently the condition is $G^*(\Phi(x)) = K(x) \cdot 2^{l(|x|)} \cdot \mathtt{ACC}_V(x)$.

Remark 2.14 More generally, we could say what it means for one optimization problem (S', G')—think of it as already shown hard to approximate—to reduce to another optimization problem (S, G). However all reductions in this paper are from the basic 2P1R optimization problem so for simplicity we state the definition for the case $(S', G') = \mathrm{OPT}_V$.

Remark 2.15 This is a strong notion of reduction since it preserves the value (up to a multiplicative factor). For some problems it is useful to consider weaker notions, but since the reductions in this paper are value preserving we will for simplicity stick to this case.

Definition 2.16 Let (S, G) be an optimization problem. We say that 2P1R reduces to (S, G), written $\mathrm{2P1R} \leq_a (S, G)$, if for every verifier V there exists a value preserving reduction of OPT_V to (S, G).

From Theorem 2.4 we get the following.

Theorem 2.17 There are constants $c, \mu > 0$ such that the following is true. Let (S, G) be an optimization problem and suppose 2P1R $\leq_a (S, G)$. Then

(1) The existence of a polynomial time, factor $2^{\Theta(\log^{1/c} n)}$ approximation algorithm for (S, G) implies $\mathrm{NP} \subseteq \bigcup_{d>0} \mathrm{DTIME}(n^{d \log^c n})$.

(2) The existence a polynomial time, factor μ approximation algorithm for (S, G) implies $\mathrm{P} = \mathrm{NP}$.

Remark 2.18 Given Remark 2.5 we can achieve $c = 3$ and $\mu = \frac{1}{2} - \epsilon_*$ in Theorem 2.17 (and Theorem 1.1), for any $\epsilon_* > 0$.

Remark 2.19 If every $L \in \mathrm{NP}$ had a 2P1R proof with complexity $O(\log n)$ and error $1/n$ then we could replace the two conclusions in Theorem 2.17 by the single conclusion saying the following: the existence of a polynomial time $n^{\Theta(1)}$ factor approximation algorithm for (S, G) implies $\mathrm{P} = \mathrm{NP}$. Theorem 1.1 would be improved correspondingly.

Remark 2.20 If 2P1R $\leq_a (S, G)$ then (a natural decision version of) the optimization problem (S, G) is NP-hard. Specifically, for every (S, G) there exist constants $d, \epsilon > 0$ and a polynomial time computable map $K\colon \{0,1\}^* \to \mathsf{R}_+^*$ such that the language

$$\begin{aligned} L &= \{x : \exists y \in S(x)\, [G(x, y) > K(x) \cdot |x|^d \cdot \epsilon]\,\} \\ &= \{x : G^*(x) > K(x) \cdot |x|^d \cdot \epsilon\} \end{aligned}$$

is NP-hard. This follows from the definitions and Theorem 2.4(2).

3 The Three Reductions

We will show that 2P1R reduces to each of the problems MAX CAPACITY REPS, MAX PRIORITY FLOW and LONGEST CR PATH (Propositions 3.1, 3.3 and 3.4 below). Theorem 1.1 then follows from Theorem 2.17.

3.1 Max Capacity Reps

The formal statement of the problem was given in Section 1.2. We can think of the points in $S_1 \cup \cdots \cup S_m$ as processors in a network; each S_i is a cluster or processors. The capacity $c(x, y)$ for points x, y in different clusters might represent the capacity of a link connecting these processors. We wish to find a set of leaders, one per cluster, such that the total link capacity in connections between the leaders is maximized.

MAX CAPACITY REPS is just a restatement of the 2P1R optimization problem in a different "language," and the reduction which we now provide is very simple.

Proposition 3.1 2P1R $\leq_a$MAX CAPACITY REPS.

Proof: Let $V = (\pi, \rho)$ be a verifier of complexity l. We construct a value preserving reduction Φ of OPT_V to MAX CAPACITY REPS.

Let $x \in \{0,1\}^n$ and write l for $l(n)$. Let $P = \{\pi^1(x, R) : R \in \{0,1\}^l\}$ and $Q = \{\pi^2(x, R) : R \in \{0,1\}^l\}$. For each $p \in P$ and $q \in Q$ let $R(p, q) = |\{R \in \{0,1\}^l : \pi(x, R) = (p, q)\}|$.

The sets in the instance $\Phi(x)$ are $A_p^1 = \{\langle 1, p, a\rangle : a \in \{0,1\}^l\}$ for each $p \in P$, and $B_q^2 = \{\langle 2, q, b\rangle : b \in \{0,1\}^l\}$ for each $q \in Q$—a total of $|P| + |Q| \leq 2^{l+1}$ sets. For each p, a, q, b we set $c(\langle 1, p, a\rangle, \langle 2, q, b\rangle) = c(\langle 2, q, b\rangle, \langle 1, p, a\rangle) = R(p, q) \cdot \rho(x, p, a, q, b)$. We set all other capacities to 0. One can check that this instance can be constructed in time polynomial in n and 2^l. Let

$c(T) = \sum_{x,y \in T} c(x,y)$ denote the capacity of a system of representatives T, and let c^* be the maximum of $c(T)$ over all systems of representatives T.

Let (A, B) be a pair of provers. We may assume wlog that $|A(p)| = |B(q)| = l$ for all $p \in P$ and $q \in Q$ (this will only increase $\mathtt{ACC}_{V,(A,B)}(x)$). Then $T = \{\langle 1, p, A(p)\rangle : p \in P\} \cup \{\langle 2, q, B(q)\rangle : q \in Q\}$ is a system of representatives of capacity $c(T) = 2 \cdot 2^l \cdot \mathtt{ACC}_{V,(A,B)}(x)$. This implies $c^* \geq 2 \cdot 2^l \cdot \mathtt{ACC}_V(x)$.

On the other hand, suppose T is a system of representatives. For each $p \in P$ we set $A(p)$ to the unique string a such that $\langle 1, p, a\rangle \in T \cap A_p^1$. For each $q \in Q$ set $B(q)$ to the unique string b such that $\langle 2, q, b\rangle \in T \cap B_q^2$. (To formally make A, B into a pair of provers, also define the former arbitrarily outside P and the latter arbitrarily outside Q. From now on we will ignore this issue). One can check that $c(T) = 2 \cdot 2^l \cdot \mathtt{ACC}_{V,(A,B)}(x)$. This implies that $2 \cdot 2^l \cdot \mathtt{ACC}_V(x) \geq c^*$. Put together with the above we have shown $c^* = 2 \cdot 2^l \cdot \mathtt{ACC}_V(x)$ which establishes that Φ is a value preserving reduction. ■

Remark 3.2 MAX CAPACITY REPS can also be thought of as a digraph problem, with the nodes corresponding to the elements of the sets and the edges carrying the capacity. In this light we note that the reduction used above guarantees certain properties; for example, we get a graph (as opposed to a digraph), and this graph is bipartite (edges of capacity 0 may simply be omitted). Specifically, it is the case that 2P1R reduces also to the following problem which is thus hard to approximate in the sense of Theorem 2.17.

MAX BIPARTITE REPS
Instance: Bipartite graph $G = (V_1, V_2, E)$. A partition of the left hand side nodes V_1 into sets $S_1^1, \ldots, S_m^1$ and a partition of the right hand side nodes V_2 into sets $S_1^2, \ldots, S_m^2$. For each edge e a capacity $c(e) \geq 0$.
Solutions: A set $T \subset V_1 \cup V_2$ is a solution if it contains exactly one member of S_i^j for each $i = 1, \ldots, m$ and each $j = 1, 2$.
Value of Solutions: The value of a solution T is $c(T) = \sum_{e \in E_T} c(e)$, where E_T is the set of edges of the subgraph induced by T.

3.2 Max Priority Flow

In the basic network flow problem we are given a digraph $G = (V, E)$, sources $s_1, \ldots, s_k \in V$, sinks $t_1, \ldots, t_l \in V$, and a capacity function c: $E \to \mathsf{R}_+$. A flow is a function f: $E \to \mathsf{R}_+$ assigning to each arc e a number $f(e) \geq 0$ which represents the amount of flow along this arc. At any vertex v we define the incoming flow as $\sum_{(u,v) \in E} f(u,v)$ and the outgoing flow as $\sum_{(v,u) \in E} f(v,u)$. Flow must be conserved at each non-source and non-sink node (that is, incoming flow must equal outgoing flow) and must also be below capacity on each arc (that is, $f(e) \leq c(e)$ for all $e \in E$). The problem of finding a max flow (one which maximizes the amount of flow entering the sinks) is solvable in polynomial time [FF]. Under various additional constraints, the problem becomes NP-hard (cf. [GJ, Appendix A2.4]).

We stated in Section 1.2 the version we consider. The priority ordering for vertex v says that if arc e_1 precedes arc e_2 in the ordering then e_1 must be filled before flow is allowed to enter e_2. We stress that $\prec_v$ is a *partial* order. We also stress that that we are measuring, as the value of the flow, the amount entering sink t_1 (rather than the sum of amounts entering $t_1, \ldots, t_l$). We now provide the reduction that establishes that our problem is hard to approximate.

Proposition 3.3 2P1R $\leq_a$MAX PRIORITY FLOW.

Proof: Let $V = (\pi, \rho)$ be a verifier of complexity l. We construct a value preserving reduction Φ of OPT_V to MAX PRIORITY FLOW.

Let $x \in \{0,1\}^n$ and write l for $l(n)$. Let $P = \{\pi^1(x,R) : R \in \{0,1\}^l\}$ and $Q = \{\pi^2(x,R) : R \in \{0,1\}^l\}$. For each $p \in P$ and $q \in Q$ let

$$\begin{aligned} R(p,q) &= |\{R \in \{0,1\}^l : \pi(x,R) = (p,q)\}| \\ R^1(p) &= |\{R \in \{0,1\}^l : \pi^1(x,R) = p\}| \\ R^2(q) &= |\{R \in \{0,1\}^l : \pi^2(x,R) = q\}| \ . \end{aligned}$$

Note $R^1(p) = \sum_{q \in Q} R(p,q)$ and $R^2(q) = \sum_{p \in P} R(p,q)$.

The vertex set of the digraph in the instance $\Phi(x)$ consists of the union of the following sets.

$$\begin{aligned} &\{s, t_1, t_2, \langle s,1\rangle, \langle s,2\rangle\} \\ &\{\langle 1,p\rangle : p \in P\} \\ &\{\langle 2,q\rangle : q \in Q\} \\ &\{\langle 1,p,a\rangle : p \in P;\ a \in \{0,1\}^l\} \\ &\{\langle 2,q,b\rangle : q \in Q;\ b \in \{0,1\}^l\} \\ &\{\langle p,a,q,b\rangle : p \in P;\ q \in Q;\ a,b \in \{0,1\}^l\} \ . \end{aligned}$$

The (unique) source is s, and t_1, t_2 are the sinks. For $i = 1,2$ and all p,a,q,b we insert arcs

$$\begin{aligned} &(s, \langle s,i\rangle) \\ &(\langle s,1\rangle, \langle 1,p\rangle) \text{ and } (\langle s,2\rangle, \langle 2,q\rangle) \\ &(\langle 1,p\rangle, \langle 1,p,a\rangle) \text{ and } (\langle 2,q\rangle, \langle 2,q,b\rangle) \\ &(\langle 1,p,a\rangle, \langle p,a,q,b\rangle) \text{ and } (\langle 2,q,b\rangle, \langle p,a,q,b\rangle) \\ &(\langle p,a,q,b\rangle, t_i) \ . \end{aligned}$$

For $i = 1,2$ and all p,a,q,b define the capacities

$$\begin{aligned} c(s, \langle s,i\rangle) &= 2^l \\ c(\langle s,1\rangle, \langle 1,p\rangle) &= R^1(p) \\ c(\langle s,2\rangle, \langle 2,q\rangle) &= R^2(q) \\ c(\langle 1,p\rangle, \langle 1,p,a\rangle) &= R^1(p) \\ c(\langle 2,q\rangle, \langle 2,q,b\rangle) &= R^2(q) \\ c(\langle 1,p,a\rangle, \langle p,a,q,b\rangle) &= R(p,q) \\ c(\langle 2,q,b\rangle, \langle p,a,q,b\rangle) &= R(p,q) \ . \end{aligned}$$

Also for all p,a,q,b define the capacities

$$\begin{aligned} c(\langle p,a,q,b\rangle, t_1) &= R(p,q) \cdot \rho(x,p,a,q,b) \\ c(\langle p,a,q,b\rangle, t_2) &= R(p,q) \cdot [2 - \rho(x,p,a,q,b)] \ . \end{aligned}$$

For each p,q define the upper bounds

$$b(\langle 1,p\rangle) = b(\langle 2,q\rangle) = 1 \ .$$

There is no upper bound for the rest of the nodes (formally, the upper bound on any other node is set to the number of its outgoing arcs). For each p,a,q,b the priority ordering for vertex $\langle p,a,q,b\rangle$ assigns to $(\langle p,a,q,b\rangle, t_2)$ a higher priority than $(\langle p,a,q,b\rangle, t_1)$; there is no priority ordering for other vertices. This completes the description of the instance $\Phi(x)$. One can check that this instance can be constructed in time polynomial in n and 2^l.

Let (A,B) be a pair of provers. We may assume wlog that $|A(p)| = |B(q)| = l$ for all $p \in P$ and $q \in Q$ (this will only increase $\mathtt{ACC}_{V,(A,B)}(x)$). For all p,a,q,b define $\delta_A(p,a) = 1$ if $a = A(p)$ and 0 otherwise; $\delta_B(q,b) = 1$ if $b = B(q)$ and 0 otherwise; and $\delta_{A,B}(p,a,q,b) = \delta_A(p,a) \cdot \delta_B(q,b)$. We

define the following flow f. For $i = 1, 2$ and all p, q set the flow equal to capacity on the following arcs: $(s, \langle s, i\rangle), (\langle s, 1\rangle, \langle 1, p\rangle), (\langle s, 2\rangle, \langle 2, q\rangle)$. For $i = 1, 2$ and all p, a, q, b set

$$\begin{aligned} f(\langle 1, p\rangle, \langle 1, p, a\rangle) &= R^1(p) \cdot \delta_A(p, a) \\ f(\langle 2, q\rangle, \langle 2, q, b\rangle) &= R^2(q) \cdot \delta_B(q, b) \\ f(\langle 1, p, a\rangle, \langle p, a, q, b\rangle) &= R(p, q) \cdot \delta_{A,B}(p, a, q, b) \\ f(\langle 2, q, b\rangle, \langle p, a, q, b\rangle) &= R(p, q) \cdot \delta_{A,B}(p, a, q, b) \ . \end{aligned}$$

Finally, for $i = 1, 2$ and all p, a, q, b set

$$f(\langle p, a, q, b\rangle, t_i) = c(\langle p, a, q, b\rangle, t_i) \cdot \delta_{A,B}(p, a, q, b) \ .$$

One can check that f is a priority flow, and that its value is $2^l \cdot \mathtt{ACC}_{V,(A,B)}(x)$. This implies that the maximum priority flow has value at least $2^l \cdot \mathtt{ACC}_V(x)$.

Now suppose f is a priority flow. For each $p \in P$ the upper bound $b(\langle 1, p\rangle) = 1$ implies that there is at most one string a such that $f(\langle 1, p\rangle, \langle 1, p, a\rangle) > 0$; define $A(p)$ to be this string if it exists, and arbitrary otherwise. Similarly for each $q \in Q$ define $B(q)$ to be the unique string b for which $f(\langle 2, q\rangle, \langle 2, q, b\rangle) > 0$ if such a string exists, and arbitrary otherwise. For all p, a, q, b, based on the fact that the flow obeys the priority ordering at $\langle p, a, q, b\rangle$, one can argue that if $(a, b) \neq (A(p), B(q))$ then $f(\langle p, a, q, b\rangle, t_1) = 0$. On the other hand for all p, a, q, b such that $(a, b) = (A(p), B(q))$ it is the case that $f(\langle p, a, q, b\rangle, t_1) \leq R(p, q) \cdot \rho(x, p, a, q, b)$. From this one can argue that the value of f is at most $2^l \cdot \mathtt{ACC}_{V,(A,B)}(x)$. We omit the details. ■

3.3 Longest CR Path

The problem was stated in Section 1.2. We now provide the reduction which shows that LONGEST CR PATH is hard to approximate.

Proposition 3.4 2P1R $\leq_a$ LONGEST CR PATH.

Proof: Let $V = (\pi, \rho)$ be a verifier of complexity l. We construct a value preserving reduction Φ of OPT_V to LONGEST CR PATH.

Let $x \in \{0,1\}^n$ and write l for $l(n)$. Let $P = \{\pi^1(x, R) : R \in \{0,1\}^l\}$ and $Q = \{\pi^2(x, R) : R \in \{0,1\}^l\}$. For each $p \in P$ and $q \in Q$ let $R(p, q) = |\{R \in \{0,1\}^l : \pi(x, R) = (p, q)\}|$. We set $N = 2^l$, and identify $\{0,1\}^l$ with $\{1, \ldots, N\}$ and $\{0,1\}^{2l}$ with $\{1, \ldots, N^2\}$ in the natural manner. We describe the construction of the graph G component by component. The first set of components is an *elimination matrix* $M(pq, \bar{p}\bar{q})$ for each $p, \bar{p} \in P$ and $q, \bar{q} \in Q$.

Component: Elimination Matrix $M(pq, \bar{p}\bar{q})$

Vertices: The vertex set is $\{\langle pq, \bar{p}\bar{q}, \bar{a}\bar{b}, ab\rangle : a, b, \bar{a}, \bar{b} \in \{0,1\}^l\}$.

Vertex Types: All nodes are monochromatic.

Red Edges: We add red edges $(\langle pq, \bar{p}\bar{q}, i, ab\rangle, \langle pq, \bar{p}\bar{q}, i+1, ab\rangle)$ for all $i = 1, \ldots, N^2 - 1$ and $a, b \in \{0,1\}^l$.

Blue Edges: For all $\bar{a}, \bar{b} \in \{0,1\}^l$ we define a set of vertices $\mathcal{E}(pq, \bar{p}\bar{q}, \bar{a}\bar{b})$ as follows: if $p = \bar{p}$ and $q \neq \bar{q}$ then it is $\{\langle pq, \bar{p}\bar{q}, \bar{a}\bar{b}, ab\rangle : a, b \in \{0,1\}^l \text{ and } a \neq \bar{a}\}$; else if $p \neq \bar{p}$ and $q = \bar{q}$ then it is $\{\langle pq, \bar{p}\bar{q}, \bar{a}\bar{b}, ab\rangle : a, b \in \{0,1\}^l \text{ and } b \neq \bar{b}\}$; else it is $\emptyset$. Let $v_j[pq, \bar{p}\bar{q}, \bar{a}\bar{b}]$ denote the j-th element (under the lexicographic ordering) of the set $\mathcal{E}(pq, \bar{p}\bar{q}, \bar{a}\bar{b})$. Now we add blue edges $(v_j[pq, \bar{p}\bar{q}, \bar{a}\bar{b}], v_{j+1}[pq, \bar{p}\bar{q}, \bar{a}\bar{b}])$ for all $j = 1, \ldots, |\mathcal{E}(pq, \bar{p}\bar{q}, \bar{a}\bar{b})| - 1$ and $\bar{a}, \bar{b} \in \{0,1\}^l$.

Weights: All edges have weight 0.

We visualize $M(pq, \bar{p}\bar{q})$ as an N^2 by N^2 matrix whose row $\bar{a}\bar{b}$ column ab element is $\langle pq, \bar{p}\bar{q}, \bar{a}\bar{b}, ab\rangle$. Red edges connect every node to the node directly beneath it in the following row. Blue edges in row $\bar{a}\bar{b}$ link all the nodes whose column number ab has $a \neq \bar{a}$ (if $p = \bar{p}$) or $b \neq \bar{b}$ (if $q = \bar{q}$). Intuitively, column ab corresponds to the provers replying a to p and b to q. Each row thus corresponds to a list of all possible responses of the provers. Assuming a response $\bar{a}$ to $\bar{p}$ and a response $\bar{b}$ to $\bar{q}$ have been selected, we will make sure that the color respecting path goes through $\mathcal{E}(pq, \bar{p}\bar{q}, \bar{a}\bar{b})$ via blue edges. Such a path then cannot go through any of the red edges in the columns which contain a vertex in $\mathcal{E}(pq, \bar{p}\bar{q}, \bar{a}\bar{b})$. Thus, such a path effectively "eliminates" the subset of columns ab corresponding to responses to pq which would be incompatible with the responses $\bar{a}, \bar{b}$ to $\bar{p}, \bar{q}$ respectively.

Component: Entrance and exit nodes and response choosing edges for $p \in P$ and $q \in Q$.

Vertices: Add a node $\langle pq, 1\rangle$ (we call this the *entrance* node for pq). Add nodes $\langle pq, 2, ab\rangle$ for all $a, b \in \{0,1\}^l$ (we call these the exit nodes for pq).

Vertex Types: All these nodes are dichromatic.

Red Edges: Add red edges $(\langle pq, 1\rangle, \langle pq, 1, 1, ab\rangle)$, $(\langle pq, N^2, N^2, ab\rangle, \langle pq, 2, ab\rangle)$, and $(\langle pq, i, N^2, ab\rangle, \langle pq, i+1, 1, ab\rangle)$ for all $i = 1, \ldots, N^2 - 1$ and $a, b \in \{0,1\}^l$.

Weights: We assign $(\langle pq, N^2, N^2, ab\rangle, \langle pq, 2, ab\rangle)$ a weight of $R(p,q) \cdot \rho(x, p, a, q, b)$, and we assign all other edges described here a weight of 0.

In other words, for each pq, we line up the collection of elimination matrices $M(pq, 1), \ldots, M(pq, N^2)$. We connect the entrance node of pq to all nodes in the first row of $M(pq, 1)$ with red edges, and assign these edges a weight proportional to the probability that the verifier asks questions p, q and then accepts p, a, q, b (where ab is the column). We link (with a red, weight 0 edge) each node of the last row of $M(pq, i)$ with the node in the same column of the first row of $M(pq, i+1)$. We also link (with a red, weight 0 edge) the last nodes of $M(pq, N^2)$ to the corresponding exit nodes of pq. We will refer to the structure just constructed (the entrance node $\langle pq, 1\rangle$, the matrices $M(pq, 1), \ldots, M(pq, N^2)$, the exit nodes $\langle pq, 2, ab\rangle$ and the links between these described above) as the *pq component* of the graph.

Component: Elimination Edges for $p \in P$ and $q \in Q$.

Blue Edges: Let $C(pq) = \{\bar{p}\bar{q} : p = \bar{p} \text{ or } q = \bar{q}\} - \{pq\}$. Let $u_j[pq]$ denote the j-th element (under the lexicographic ordering) of $C(pq)$. Note that $|\mathcal{E}(\bar{p}\bar{q}, pq, ab)|$ is independent of $\bar{p}\bar{q}, pq, ab$ whenever $\bar{p}\bar{q} \in C(pq)$; call this number m. For all $j = 1, \ldots, 2N-2$ and $a, b \in \{0,1\}^l$ we add blue edges as follows. First, we add $(\langle pq, 2, ab\rangle, \langle u_1[pq], pq, ab, v_1[u_1[pq], pq, ab]\rangle)$. If $pq < N^2$ we also add $(\langle u_{2N-1}[pq], pq, ab, v_m[u_{2N-1}[pq], pq, ab]\rangle, \langle pq+1, 1\rangle)$. Finally we add $(\langle u_j[pq], pq, ab, v_m[u_j[pq], pq, ab]\rangle, \langle u_{j+1}[pq], pq, ab, v_1[u_{j+1}[pq], pq, ab]\rangle)$.

Weights: All these edges have weight 0.

Here we consider the elimination matrices $M(\bar{p}\bar{q}, pq)$ for which either $\bar{p} = p$ or $\bar{q} = q$. We pick a row ab, and look at the chain of blue edges at this row in each of these matrices. We hook these chains together by connecting the last element of this chain on one matrix to the first on the chain in the next matrix. We also connect the exit node of the ab column of pq to the first element of the chain of the first matrix, and the last element of the chain of the last matrix to the entrance node of the $pq+1$ component.

This completes the description of the instance $\Phi(x)$. One can check that this instance can be constructed in time polynomial in n and 2^l. We now try to provide a rough idea of why Φ is value preserving.

Given a pair (A, B) of provers, we can define a CR path in G as follows. We begin at $\langle 1, 1\rangle$. At a node of the form $\langle pq, 1\rangle$ we take the (red) edge $(\langle pq, 1\rangle, \langle pq, 1, 1, A(p)B(q)\rangle)$, and at all other vertices we "follow our nose" (choices of edges are uniquely defined by the colors of edges and types of vertices). We end at $\langle N^2, 2, A(N)B(N)\rangle$. One can check that this (CR) path has weight equal to $2^l \cdot \mathtt{ACC}_{V,(A,B)}(x)$.

On the other hand let T be a longest CR path in G. We assume that T starts at $\langle 1, 1\rangle$ and ends at $\langle N^2, 2, ab\rangle$ for some ab (this is without loss of generality). We claim that if for some $p, \bar{p}, q, \bar{q}$, $a, \bar{a}, b, \bar{b}$ both $(\langle pq, 1\rangle, \langle pq, 1, 1, ab\rangle)$ and $(\langle \bar{p}\bar{q}, 1\rangle, \langle \bar{p}\bar{q}, 1, 1, \bar{a}\bar{b}\rangle)$ are on T then $p = \bar{p}$ implies $a = \bar{a}$ and $q = \bar{q}$ implies $b = \bar{b}$. The reason is that if T goes through $(\langle pq, 1\rangle, \langle pq, 1, 1, ab\rangle)$ then it must go through $\mathcal{E}(\bar{p}\bar{q}, pq, ab)$ via blue edges, and thus, to reach an exit node of the $\bar{p}\bar{q}$ component, it must go through columns of this component different from those in $\mathcal{E}(\bar{p}\bar{q}, pq, ab)$. Given this, we can associate a pair (A, B) of provers to the path. Namely, for each p, q take the (unique) edge of the form $(\langle pq, 1\rangle, \langle pq, 1, 1, ab\rangle)$ on the path and set $A(p) = a$ and $B(q) = b$. Moreover, the weight of the path equals $2^l \cdot \mathtt{ACC}_{V,(A,B)}(x)$. ■

Acknowledgments

Silvio Micali suggested looking at the longest path problem, and also contributed to the proof of Proposition 3.4; I thank him for both. I am grateful to Shafi Goldwasser for many discussions and insights on the subject of interactive proofs and approximation. Remark 2.6 was realized together with Shafi and Alex Russell. I am grateful to Oded Goldreich for comments on the paper and its presentation. Thanks to David Zuckerman for discussions on these reductions.

Work done while the author was at the IBM T.J. Watson Research Center, New York.

References

[AS] S. Arora and S. Safra. Approximating Clique is NP-Complete. *Proceedings of the 33rd Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1992).

[ALMSS] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof Verification and Intractability of Approximation Problems. *Proceedings of the 33rd Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1992).

[BFLS] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking Computations in Polylogarithmic Time. *Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing,* ACM (1991).

[BFL] L. Babai, L. Fortnow and C. Lund. Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. Computational Complexity **1**, 3–40.

[Be] M. Bellare. Interactive Proofs and Approximation. IBM Research Report RC 17969 (May 1992).

[BGLR] M. Bellare, S. Goldwasser, C. Lund and A. Russell. Efficient Probabilistically Checkable Poofs and Applications to Approximation. *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing,* ACM (1993).

[BR] M. Bellare and P. Rogaway. The Complexity of Approximating a Nonlinear program. *Complexity of Numerical Optimization*, Ed. P.M. Pardalos, World Scientific (1993). Preliminary version appeared as IBM Research Report RC 17831 (March 1992).

[Co] A. Condon. The Complexity of the Max Word Problem, or the Power of One-Way Interactive Proof Systems. STACS 91.

[BGKW] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions. *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing,* ACM (1988).

[FF] L. Ford and D. Fulkerson. Flows in Networks. Princeton University Press, Princeton NJ (1962).

[FRS] L. Fortnow, J. Rompel and M. Sipser. On the Power of Multiprover Interactive Protocols. Structures 88.

[Fe] U. Feige. On the Success Probability of the Two Provers in One Round Proof Systems. Structures 1991.

[FGLSS] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating Clique is almost NP-Complete. *Proceedings of the 32nd Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1991).

[FL] U. Feige and L. Lovász. Two-Prover One Round Proof Systems: Their Power and their Problems. *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing,* ACM (1992).

[GJ] M. Garey and D. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, 1979.

[KMR] D. Karger, R. Motwani and G.D.S. Ramkumar. On Approximating the Longest Path in a Graph. Manuscript.

[Ki] J. Kilian. Strong Separation Models of Multi-Prover Interactive Proofs. Manuscript. Presented at Dimacs Workshop on Cryptography, Princeton (October 1990).

[LS] D. Lapidot and A. Shamir. Fully Parallelized Multi-Prover Protocols for NEXP-time. *Proceedings of the 32nd Annual IEEE Symposium on the Foundations of Computer Science,* IEEE (1991).

[LY1] C. Lund and M. Yannakakis. On the Hardness of Approximating Minimization Problems. *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing,* ACM (1993).

[LY2] C. Lund and M. Yannakakis. The Approximation of Maximum Subgraph Problems. ICALP 93.

[Zu] D. Zuckerman. NP-Complete Problems have a Version that is Hard to Approximate. Structures 93.