Approximation Algorithms for Unique Games

Luca Trevisan[*]
U.C. Berkeley, Computer Science Division
`luca@cs.berkeley.edu`

Abstract

We present a polynomial time algorithm based on semidefinite programming that, given a unique game of value $1 - O(1/\log n)$, satisfies a constant fraction of constraints, where n is the number of variables. For sufficiently large alphabets, it improves an algorithm of Khot (STOC'02) that satisfies a constant fraction of constraints in unique games of value $1 - O(1/(k^{10}(\log k)^5))$, where k is the size of the alphabet. We also present a simpler algorithm for the special case of unique games with linear *constraints.*

Finally, we present a simple approximation algorithm for 2-to-1 *games.*

1. Introduction

A *unique game* [FL92] is described by a set V of variables, taking values over a set S, the *alphabet* of the game, and a collection of constraints of the form

$$v = f(u)$$

where $u, v \in V$ are variables and $f : S \to S$ is a permutation. We are interested in finding the assignment $A : V \to S$ to the variables that satisfies the largest number of constraints. The *value* of a unique game is the fraction of constraints satisfied by an optimal assignment. For example, the following is a unique game with $V = \{v_1, v_2, v_3, v_4\}$ and $S = \{a, b, c\}$:

$$\begin{aligned} v_3 &= \begin{pmatrix} a & b & c \\ c & b & a \end{pmatrix}(v_1) \\ v_3 &= \begin{pmatrix} a & b & c \\ a & c & b \end{pmatrix}(v_2) \\ v_1 &= v_2 \\ v_4 &= \begin{pmatrix} a & b & c \\ b & c & a \end{pmatrix}(v_2) \end{aligned}$$

Such a system of constraints is unsatisfiable (in particular, it is impossible to simultaneously satisfy the first three constraints), but the assignment $(v_1, v_2, v_3, v_4) = (c, a, a, b)$ satisfies three constraints. The value of the game, therefore, is $3/4$.

We will adopt the convention that if $v = f(u)$ is a constraint, then $v > u$ in lexicographic order, a convention that is made with no loss of generality because the constraint $u = g(v)$ with $u < v$ is equivalent to $v = g^{(-1)}(u)$. We allow the same pair of variables to occur in more than one constraint.

The *constraint graph* of a unique game is the graph $G = (V, E)$ where V is the set of variables and there is an edge $e = (u, v)$ in E for every constraint $v = f(u)$. If the same two variables occur in multiple constraints, then G is a graph with parallel edges.

Formally, a unique game will be a triple $(G = (V, E), \{f_e\}_{e \in E}, S)$ where G is the constraint graph, S is the range of the variables, and for every edge $e \in E$, $e = (u, v)$, $v > u$, we have the constraint $v = f_{v,u}(u)$.

Clearly, given a unique game, if there exists an assignment that satisfies all constraints then it is easy to find such an assignment. One may assume without loss of generality that the graph is connected (otherwise, apply the following algorithm to each connected component), and so one can find a spanning tree. Let A be a satisfying assignment for the unique game: guess the value $A(r)$,

* This paper is based upon work supported by the National Science Foundation under grant CCF 0515231 and by the US-Israel Binational Science Foundation under grant 2002246.

where r is the root of the spanning tree, and then find $A(v)$ for every other vertex v, by noting that $A(v) = f_{v,u_k}(f_{u_k,u_{k-1}}(\cdots(f_{u_2,u_1}(f_{u_1,r}(A(r))))\cdots))$ where $r, u_1, \ldots, u_k, v$ is the path from r to v in the spanning tree.[1]

If one is given an *almost* satisfiable unique game, that is, a game admitting an assignment that satisfies a $1-\gamma$ fraction of constraint for some small γ, then it is not clear how to find a good assignment. Khot's [Kho02] unique games conjecture is that for every $\gamma > 0$ there is a constant $k = k(\gamma)$ such that it is hard to distinguish games of value $\geq 1-\gamma$ from games of value $\leq \gamma$, even when restricted to alphabets of size k and to graphs that are bipartite.

The unique games conjecture has been shown to imply several inapproximability results, such as that the minimum edge-deletion bipartite subgraph problem is hard to approximate within any constant [Kho02], that the Vertex Cover problem is hard to approximate within $2-\varepsilon$ [KR03], that the Max Cut problem is hard to approximate within $.878\cdots$ [KKMO04, MOO05], and that the Sparsest Cut problem [CKK+05] is hard to approximate within any constant. Extensions of the unique games conjecture to sublinear γ has been used to prove that the Sparsest Cut problem is hard to approximate within a $\Omega((\log\log n)^{1/6})$ [KV05] and within a $\Omega(\log\log n)$ factor [CKK+05].

In partial support of this conjecture, Feige and Reichman [FR04] prove that it is hard to approximate the value of a unique game within a factor $2^{(\log n)^{.99}}$, where n is the number of constraints. The value of the instances produced by their reduction, however, is very small, and so their result does not show that it is hard to distinguish instances of value $1-\gamma$ from instances of value γ.

Given that there are now several results [Kho02, KR03, KKMO04, MOO05, KV05, CKK+05] based on the unique games conjecture, there is a good motivation to investigate algorithms that might contradict it. We are aware of only one algorithmic result, due to Khot [Kho02], for unique games. Khot's algorithm shows that given a unique game with an alphabet of size k and the promise that a $1 - O(\varepsilon^5/(k^{10}(\log k)^5)$ constraint can be satisfied, it is possible to efficiently find an assignment that satisfies a $1-\varepsilon$ fraction of constraints.

Khot [Kho02] also introduces *2-to-1 games*. In general, for integers d, d' a d-to-d' game is defined by a set of variables V ranging over a set S, and a set of constraints of the form

$$P(u,v) = 1$$

where $u, v \in V$ are variables and $P : S \times S \to \{0,1\}$ is a d-to-d' predicate, that is, a predicate such that for every $a \in S$ there are at most d' values b such that $P(a,b) = 1$, and for every $b \in S$ there are at most d values a such that $P(a,b) = 1$. As before, we want to find an assignment $A : V \to S$ to the variables that maximizes the number of satisfied constraints. The *value* of a d-to-d' game is the fraction of constraints satisfied by an optimal assignment. We will restrict ourselves to the case $d = d'$, which is done without loss of generality.[2] In a d-to-d game we will follow the convention that if $P(u,v) = 1$ is a constraint then $v > u$. As before, the constraint graph of a game is a graph $G = (V,E)$ where V is the set of variables and E contains an undirected edge for every constraint. A d-to-d game is formally specified by a triple $(G = (V,E), \{P_e\}, S)$ where $P_e : S \times S \to \{0,1\}$ are d-to-d predicates.[3]

A unique game is a 1-to-1 game. An example of a 2-to-2 game is the 3-coloring problem. If G is a graph, consider the 2-to-2 game $(G = (V,E), \{neq_e\}_{e\in E}, \{a,b,c\})$, where the predicate $neq_e(x,y)$ is satisfied if and only if $x \neq y$. Then assignments correspond to 3-colorings, and the game has value one if and only if the graph is 3-colorable.

Khot [Kho02] conjectures that for every $\gamma > 0$ there is an alphabet S of size depending only on γ such that it is hard to distinguish satisfiable 2-to-1 games (that is, games of value 1) from games of value $\leq \gamma$, even if the games are restricted to use the fixed alphabet S. This conjecture has been shown to imply intractability results for graph coloring [DMR05]. We are aware of no previous algorithmic work on this conjecture.

1 To be precise, for an edge (u,v), $v > u$, so far we have only defined the permutation $f_{v,u}$. The permutation $f_{u,v}$ is defined to be $f_{u,v} := f_{v,u}^{(-1)}$.

2 A d-to-d' game is also a $\max\{d,d'\}$-to-$\max\{d,d'\}$ game for a stronger reason.

3 The definition of 2-to-1 game given in [Kho02] is slightly different, and the one given here is more general. Since we are looking for algorithms, the additional generality in the definition only makes our results stronger.

1.1. Our Results

In this paper we rule out a generalization of the unique games conjecture to the case of sublinear γ that could have been considered plausible given the result of Feige and Reichman [FR04].

Theorem 1 *There is a constant c and an algorithm that, on input a unique game $(G = (V, E), \{f_e\}, S)$, a parameter ε, and the promise that the value of the game is at least $1 - c\varepsilon^3/\log|E|$, returns an assignment that satisfies at least a $1 - \varepsilon$ fraction of constraints. The algorithms work in time polynomial in $|E|$ and $|S|$.*

The algorithm of Theorem 1, described in Section 3 works by solving and then rounding a semidefinite programming (SDP) relaxation of the unique game.[4] The same algorithmic ideas can be used to achieve different trade-offs. For example, we can show that given a game of value $1-\delta$ we can satisfy, in polynomial time, a $1/|E|^{O(\delta)}$ fraction of constraints.

We prove a stronger result for linear unique games, that is, for unique games such that each constraint is a linear equation. Our algorithm works even with alphabets of exponential size.

Theorem 2 *There is a constant c and an algorithm that, on input a linear unique game $(G = (V, E), \{f_e\}, S)$, a parameter ε, and the promise that the value of the game is at least $1 - c\varepsilon^2/\log|E|$, returns an assignment that satisfies at least a $1-\varepsilon$ fraction of constraints. The algorithms work in time polynomial in $|E|$ and $\log|S|$.*

For 2-to-1 games we are able to prove a weaker result, although our algorithm is sufficient to rule out the type of hardness result that is known for general 2-variable constraint satisfaction problems [Raz98]. Our algorithm also works for d-to-d games for large (even super-constant) d.

Theorem 3 *There is an algorithm that, on input a satisfiable d-to-d game $(G = (V, E), \{P_e\}, S)$ returns an assignment that satisfies at least a $1/2^{O(\sqrt{(\log|E|)\cdot(\log d)})}$ fraction of constraints, where The algorithm runs in time polynomial in $|E|$ and $|S|$.*

4 In the full version of this paper [Tre05] we also describe a completely different, and more combinatorial, algorithm that, unfortunately, achieves a worse performance.

In particular, Theorem 3 implies that for every satisfiable d-to-d' game we can satisfy at least a $1/2^{O(\sqrt{(\log|E|)\cdot(\log\max\{d,d'\})})}$ fraction of constraints.

1.2. Overview of the Algorithms

Our results all share the same general structure: (i) we first study "nice" instances, where the constraint graph has low radius and, possibly, other conditions apply; and (ii) we show how to reduce general instances to nice ones. A result of Leighton and Rao on how to decompose general graphs into low-radius components is an important ingredient of the reductions.

SDP-Based Algorithm for General Unique Games Our approximation algorithm for general unique games is based on a semidefinite programming (SDP) relaxation of unique games. In our analysis, we first present a rounding scheme that works well in "nice" instances where the constraint graph has small radius and where every edge gives a large contribution to the objective function. We then give a reduction from the general case to the "nice" case.

Our main result is a rounding algorithm with the following guarantee: given a solution for the SDP such that every edge contributes at least $1-\gamma$ to the objective function, and assuming that the constraint graph has radius d, we can find in polynomial time a solution that satisfies at least a $1-O(d\gamma)$ fraction of constraints.

Given a general unique game of cost $1-\varepsilon^3/\log n$, we first remove all edges that contribute less than $\varepsilon^2/\log n$ to the objective function. By Markov's inequality, this step removes at most an ε fraction of edges. Then we use a decomposition procedure by Leigthon and Rao that shows how to decompose a graph in components of radius $d = O(\varepsilon^{-1}\log n)$ by removing only an ε fraction of edges. Finally, we apply our rounding procedure to each component, thus satisfying at least a $1 - O(d\cdot\varepsilon^2/\log n) = 1 - O(\varepsilon)$ fraction of edges in each component. Overall, we have satisfied at least a $1 - O(\varepsilon)$ fraction of edges.

Algorithm for Linear Unique Games For linear unique games we can do better by using in a simpler way the Leighton-Rao decomposition into low-radius components.

We first show that, if the constraint graph has radius k, we can either find an assignment that satisfies all edges within the component, or find a set of at most $4k+2$ constraints that cannot be simultaneously satisfied.

With this preliminary result in place, our algorithm works as follows: compute a decomposition of radius $k = O(\log n)$, and either satisfy all constraints within all components, or find an inconsistent set of at most $4k+2$ constraints. In the former case, stop, in the latter case remove the constraint from the graph and re-compute the decomposition.

Suppose that after T steps the algorithm stops. Then the algorithm satisfies all constraints except at most $\varepsilon|E| + T(4k+2)$ constraints. On the other hand, every assignment must contradict at least T constraints, and if we are promised that there is an assignment that satisfies at least a $1-\varepsilon/(4k+2)$ fraction of constraints it follows that $T \leq \varepsilon|E|/(4k+2)$ and so the algorithm satisfies at least a $1-2\varepsilon$ fraction of constraints.

Algorithm for 2-to-1 Games For 2-to-1 games, we start again from the case of low-radius instances and prove that, if the constraint graph has radius k and the instance is satisfiable, then it is possible to satisfy a $1/2^{2k}$ fraction of constraints in polynomial time.

If we are given a satisfiable 2-to-1 games with an arbitrary constraint graph, we can find a low-radius decomposition of the constraint graph such that each component has radius at most $O(\sqrt{\log n})$. This can be done by removing at most a $1-1/2^{O(\sqrt{\log n})}$ fraction of edges. Since we started from a satisfiable instance, each component in the decomposition is still satisfiable, and so we can satisfy at least a $1/2^{O(\sqrt{\log n})}$ fraction of edges in each component.

If we have a d-to-d game, we find a decomposition of radius $k = O(\sqrt{(\log n)/(\log d)})$, which can be done after removing at most a $1-1/2^{\sqrt{(\log n)(\log d)}}$ fraction of edges. In each component, after guessing $A(r)$ where r is the root of a depth-t spanning tree, we compute lists as above, with each list being of size at most $d^t = 2^{\sqrt{(\log n)(\log d)}}$. On average, at least a $1/2^{O(\sqrt{(\log n)(\log d)})}$ fraction of constraints is satisfied.

1.3. Weighted Version of the Problem

Recall that we allow multiple constraints to involve the same pair of variables, and so we allow the constraint graph to have multiple edges.

When we refer to the *degree* of a vertex, we mean the number of edges incident on the vertex, counting multiplicities. Similarly, we count multiplicities when we refer to the number of edges crossing a cut in a graph, or to the number of edges in an induced subgraph, and so on.

The *weighted* version of the problem, where each constraint is allowed to have an arbitrary weight, can be reduced to the unweighted version we consider in this paper by using a standard scaling and rounding approximation-preserving reduction.

Standard sparsification techniques could also be used to reduce to the case $|E| = |V| \cdot (\log |V|)^{O(1)}$, so that the dependencies on $|E|$ on the quality of the approximation could be replaced by analogous dependencies on $|V|$.

1.4. Implications of Our Results

We hope that our work will lead to further algorithmic study of unique games, and that some of the intuition that we develop in this paper will be of further use.

Our results shows that general 2-provers 1-round are more powerful than unique games, 2-to-1 or even $(\log n)^{o(1)}$-to-$(\log n)^{o(1)}$ games. (Where n is the number of constraints.) A proof of the unique games conjecture, therefore, cannot come by a direct reduction from 2-provers 1-round, except perhaps if the running time of the reduction is exponential in $1/\gamma$.

A strong form of the unique games conjecture stated in [CKK+05] is that one can have $\gamma = 1/|S|^{\Omega(1)}$ in the conjecture, for $|S| = O(\log n)$. Our result shows that the strong from cannot be true when $|S| = (\log n)^{\omega(1)}$, and so a hyptothetical proof of the [CKK+05] statement would have to distinguish between the case of small and large alphabets.

We remark that our results do not yet put any significant limitation to the possible use of unique games to prove inapproximability results. This is because the “Long Code,” used in inapproximability reductions from unique games, increases the size of the instance by at least a $2^{|S|}$ factor, and so it is not useful to have unique games with $|S| >> \log n$. When $|S| = O(\log n)$, however, it is trivial to satisfy a $1/O(\log n)$ fraction of constraints, and so one cannot have γ smaller than $1/O(\log n)$ in the conjecture.

1.5. A Later Result

Very recently Charikar, Makarychev, and Makarychev [CMM05] have devised an improved SDP-based approximation algorithm for unique games. Given a unique game of value $1-\delta$, their algorithm satisfies a $1/|S|^{O(\delta)}$ fraction of constraints. In particular, their algorithm satisfies a constant fraction of constraints in a unique game of value $1 - O(1/\log|S|)$, and thus contradicts the [CKK$^+$05] statement mentioned above.

In an instance of value $1-\delta$, our techniques only give an efficient algorithm that satisfies a $1/n^{O(\delta)}$ fraction of constraints. In the interesting case $S = O(\log n)$, the approximation of the algorithm of Charikar *et al.* is exponentially better than ours.

2. Graph Decomposition

We say that a graph $G = (V, E)$ has radius at most d if there is a vertex $r \in V$ such that every vertex $v \in V$ has distance at most d from r.

The following result is due to Leighton and Rao [LR99] and is used in several places in this paper.

Lemma 4 (Leighton and Rao [LR99]) *There is a polynomial time algorithm that, on input a graph $G = (V, E)$ and a parameter $t > 1$, returns a subset of edges $E' \subseteq E$ such that $|E|' \geq |E|/t$ and such that every connected component of the graph $G' = (V, E')$ has radius at most* $(1 + \log|E|)/(\log t)$.

PROOF: For a vertex v and an integer i, we define the *ball* $B(v, i)$ of center v and radius i as the set of vertices whose distance from v is at most i.

Consider the following algorithm, where we set $d := 1 + (\log|E|)/(\log t)$

```
1. Initialize $E' := E$
2. While there are connected components in $G' = (V, E')$ of radius larger than $d$
   (a) Let $v$ be a vertex in one such component
   (b) i:=0
   (c) While the number of edges in the cut $B(v,i), V - B(v,i)$ is larger than $(t-1)$ times the number of edges within $B(v,i)$
       • i:= i+1
   (d) Remove from $E'$ the edges in the cut $B(v,i), V - B(v,i)$
```

The main observation in the analysis of the algorithm is that the subgraph induced by $B(v, i)$ at Step 2d contains at least t^{i-1} edges, because, $B(v, 1)$ contains at least one edge and i is increased only if it makes the larger ball have at least t times as many vertices in the ball of smaller radius. This implies that the radius i of the subgraph induced by $B(v, i)$ at Step 2d is at most $1 + (\log|E|)/(\log t)$, that is, at most d, and so no vertex of that subgraph will ever be selected again at Step 2a. The edges that are removed from E' to disconnect $B(v, i)$ at Step 2d can be charged to the edges within $B(v, i)$, showing that at least a $1/t$ fraction of edges are in E'. □

3. The SDP-Based Algorithm for General Unique Games

After describing the semidefinite programming relaxation, our strategy will be to first present a rounding scheme that works well in "nice" instances where the constraint graph has small radius and where every edge gives a large contribution to the objective function. We then give a reduction from the general case to the "nice" case. The reduction uses the Leighton-Rao decomposition theorem to decompose a general graph into components of small radius.

3.1. The Relaxation

We consider the following integer programming formulation. We assume without loss of generality that the set S equals $\{1, \ldots, |S|\}$. For every variable u of the unique game we have $k := |S|$ boolean variables $u_1, \ldots, u_k$ in the integer program, with the intended meaning that if $u = i$ then $u_i = 1$ and $u_j = 0$ for $j \neq i$. Each constraint $v = f(u)$ contributes $\sum_{i \in S} v_{f(i)} u_i$ to the objective function. The integer program, therefore, looks like this:

$$\begin{array}{ll} \multicolumn{2}{l}{\max \sum_{(u,v)\in E} \sum_{i \in S} v_{\pi_{u,v}(i)} u_i} \\ \multicolumn{2}{l}{\text{Subject to}} \\ \quad u_i \cdot u_j = 0 & (\forall u \in V, \forall i, j \in [k], i \neq j) \\ \quad \sum_{i \in S} u_i = 1 & (\forall u \in V) \\ \quad u_i \in \{0, 1\} & (\forall u \in V) \end{array}$$

In the semidefinite relaxation, each variable u_i is replaced by a vector $\mathbf{u}_i$.

$\max \sum_{(u,v)\in E} \sum_{i\in S} \mathbf{v}_{\pi_{u,v}(i)} \mathbf{u}_i$
Subject to
$\mathbf{u}_i \cdot \mathbf{u}_j = 0 \qquad (\forall u \in V, \forall i,j \in [k], i \neq j)$
$\sum_{i\in S} ||\mathbf{u}_i||^2 = 1 \qquad (\forall u \in V)$

We will work with an equivalent formulation of the objective function, and add "triangle inequalities," see the SDP (1) in Figure 1.

Feige and Lovasz [FL92] and Khot [Kho02] add different inequalities, and it is not clear if the SDP (1) is equivalent or not to the ones in [FL92, Kho02].

3.2. The Rounding Procedure: Analysis on "Nice" Instances

The following lemma gives our main result about rounding the SDP (1).

Lemma 5 *Suppose we are given a unique game $(G, [k], \{\pi_e\})$ such that the SDP (1) has a feasible solution in which every edge contributes at least $1-\varepsilon/8(d+1)$, where d is the radius of the graph. Then it is possible, in polynomial time, to find a solution that satisfies a $1-\varepsilon$ fraction of the constraints.*

PROOF: We fix a spanning tree of radius d of G and we let r be the root of the tree. We pick at random a value $i \in [k]$ with probability $||\mathbf{r}_i||^2$, and we assign i to r. For every other variable v, we assign to v the value j that minimizes the "distance" $||\mathbf{v}_j - \mathbf{r}_i||^2$. Let A be the random variable corresponding to the above described distribution of assignments. We claim that every constraint has a probability at least $1-\varepsilon$ of being satisfied by such an assignment.

Let (u,v) be a constraint in G, and let $r = u^0, u^1, \ldots, u^t = u$ be a path from r to u of length $t \leq d$ in G. Let π_u be the composition of the permutations $\pi_{(r,u^1)}, \ldots, \pi_{(u^{t-1},u)}$ corresponding to the path. Let $\pi_v() := \pi_{(u,v)}(\pi_u())$.

We will show that there is a probability at least $1-\varepsilon/2$ that $A(u) = \pi_u(A(r))$ and a probability at least $1-\varepsilon/2$ that $A(v) = \pi_v(A(r))$. (We only prove the former statement, since the latter has an identical proof.) By a union bound, it will follow that there is a probability at least $1-\varepsilon$ that $A(v) = \pi_{(u,v)}(A(u))$.

By the triangle inequality, we have

$$\sum_{i\in S} ||\mathbf{r}_i - \mathbf{u}_{\pi_u(i)}||^2 \leq \varepsilon/4$$

Let B be the set of indices i such that $\pi_u(i)$ is not the j that minimizes $||\mathbf{r}_i - \mathbf{u}_j||^2$. Then we have

$$\mathbf{Pr}[A(u) \neq \pi_u(A(r))] = \sum_{i\in B} ||\mathbf{r}_i||^2$$

We claim that for every $i \in B$, $||\mathbf{r}_i - \mathbf{u}_{\pi_u(i)}||^2 \geq \frac{1}{2}||\mathbf{r}_i||^2$. This follows from the following simple fact (substitute $\mathbf{r} \leftarrow \mathbf{r}_i$, $\mathbf{u} \leftarrow \mathbf{u}_{\pi_u(i)}$ and $\mathbf{v} \leftarrow \mathbf{u}_j$, where j is the minimizer of $||\mathbf{r}_i - \mathbf{u}_j||^2$).

Claim 6 *Let $\mathbf{r}, \mathbf{u}, \mathbf{v}$ be vectors such that: (i) $\mathbf{u}\cdot\mathbf{v} = 0$, (ii) $||\mathbf{r}-\mathbf{u}||^2 \geq ||\mathbf{r}-\mathbf{v}||^2$, and (iii) the vectors $\mathbf{r}, \mathbf{u}, \mathbf{v}$ satisfy the "triangle inequality" constraints of (1).*

Then $||\mathbf{r}-\mathbf{u}||^2 \geq ||\mathbf{r}||^2$.

PROOF: We consider three cases:

1. If $||\mathbf{u}||^2 \leq \frac{1}{2}||\mathbf{r}||^2$, then
$$||\mathbf{r}-\mathbf{u}||^2 \geq ||\mathbf{r}||^2 - ||\mathbf{u}||^2 \geq \frac{1}{2}||\mathbf{r}||^2$$
2. If $||\mathbf{v}||^2 \leq \frac{1}{2}||\mathbf{r}||^2$, then
$$||\mathbf{r}-\mathbf{u}||^2 \geq ||\mathbf{r}-\mathbf{v}||^2 \geq ||\mathbf{r}||^2 - ||\mathbf{v}||^2 \geq \frac{1}{2}||\mathbf{r}||^2$$
3. If $||\mathbf{u}||^2, ||\mathbf{v}||^2 \geq \frac{1}{2}||\mathbf{r}||^2$, then from the triangle inequality and from assumption (ii) we have
$$||\mathbf{v}-\mathbf{u}||^2 \leq ||\mathbf{v}-\mathbf{r}||^2 + ||\mathbf{r}-\mathbf{u}||^2 \leq 2||\mathbf{r}-\mathbf{u}||^2$$
and by Pythagoras theorem and the orthogonality of $\mathbf{v}$ and $\mathbf{u}$ we have
$$||\mathbf{v}-\mathbf{u}||^2 = ||\mathbf{v}||^2 + ||\mathbf{u}||^2$$
so that
$$||\mathbf{r}-\mathbf{u}||^2 \geq \frac{1}{2}||\mathbf{v}-\mathbf{u}||^2 = \frac{1}{2}||\mathbf{v}||^2 + \frac{1}{2}||\mathbf{u}||^2 \geq \frac{1}{2}||\mathbf{r}||^2$$

□

We can now estimate the probability of an inconsistent assignment to $\mathbf{r}$ and $\mathbf{u}$ as

$$\begin{aligned}\mathbf{Pr}[A(u) \neq \pi_u(A(r))] &= \sum_{i\in B} ||\mathbf{r}_i||^2 \\ &\leq 2\sum_{i\in B} ||\mathbf{r}_i - \mathbf{u}_{\pi_u(i)}||^2 \\ &\leq 2\sum_{i\in S} ||\mathbf{r}_i - \mathbf{u}_{\pi_u(i)}||^2 \\ &\leq \frac{\varepsilon}{2}\end{aligned}$$

□

$$\begin{array}{ll}
\max \sum_{(u,v)\in E} \left(1 - \sum_{i\in S} \frac{1}{2}||\mathbf{v}_{\pi_{u,v}(i)} - \mathbf{u}_i||^2\right) & \\
\text{Subject to} & \\
\quad \mathbf{u}_i \cdot \mathbf{u}_j = 0 & (\forall u \in V, \forall i,j \in [k], i \neq j) \\
\quad \sum_{i\in S} ||\mathbf{u}_i||^2 = 1 & (\forall u \in V) \\
\quad ||\mathbf{w}_h - \mathbf{u}_i||^2 \leq ||\mathbf{w}_h - \mathbf{v}_j||^2 + ||\mathbf{v}_j - \mathbf{u}_i||^2 & (\forall u,v,w \in V, \forall i,j,h \in [k]) \\
\quad ||\mathbf{v}_j - \mathbf{u}_i||^2 \geq ||\mathbf{v}_j||^2 - ||\mathbf{u}_i||^2 & (\forall u,v \in V, \forall i,j \in [k])
\end{array} \qquad (1)$$

Figure 1. Semidefinite programming relaxation of a unique game.

3.3. The General Case: Proof of Theorem 1

In this section we prove the following result.

Theorem 7 *Suppose that the SDP (1) has a solution of cost at least* $(1 - c\varepsilon^3/(\log n))|E|$. *Then it is possible to find in polynomial time a solution for the unique game that satisfies at least a* $1 - \varepsilon$ *fraction of constraints. (c is an absolute constant.)*

Suppose that the SDP relaxation of a unique game $(G = (V,E), S, \{\pi_e\}_{e\in E})$ has a solution of cost at least $(1-\gamma)\cdot|E|$, where $\gamma = c\varepsilon^3/\log n$ (we will fix c later). Then for all but an $\varepsilon/3$ fraction of constraints their contribution to the objective function is at least $1 - 3\gamma/\varepsilon = 1 - 3c\varepsilon^2/\log n$. The algorithm of Theorem 7 is as follows:

1. Remove the constraints whose contribution is smaller than $1 - 3\gamma/\varepsilon$.

   This step removes at most $|E|\varepsilon/3$ edges and, in the residual graph, every edge contributes at least $1 - 3c\varepsilon^2/\log n$ to the objective function.

2. Apply the Leighton-Rao decomposition of Lemma 4 with $t = 1/(1 - \varepsilon/3))$ to the residual graph.

   This step removes at most $|E|\varepsilon/3$ edges, and the residual graph breaks up into connected components of radius at most $d = O((\log n)/\varepsilon)$.

3. Use Lemma 5 below to satisfy at least $1 - \varepsilon/3$ fraction of constraints in each connected component of the residual graph that we obtain after steps (1) and (2).

   (The constant c will have to be set so that $1 - 3c\varepsilon^2/(\log n) \geq 1 - \varepsilon/24(d+1)$, that is, $c < (\log n)/(72\cdot(d+1)\cdot\varepsilon)$.)

   This step finds a solution that satisfies all but at most $\varepsilon|E|/3$ constraints of the residual graph obtained after steps (1) and (2).

This completes the proof of Theorem 7 and Theorem 1.

It is also possible to set the parameters differently and to obtain different trade-offs. For example, we can show that given a unique game whose SDP optimum is $(1-\delta)|E|$ we can satisfy in polynomial time a $1/n^{O(\delta)}$ fraction of constraints. The above algorithm can be modified to handle this case as follows: in the first step, remove the edges that contribute less than $1 - \delta/2$ to the objective function; In the second step, find in the residual graph a decomposition of radius, say, $\frac{1}{100\delta}$ in which at least a $1/n^{O(\delta)}$ fraction of edges are not removed.

4. Linear Unique Games

We begin with the following preliminary result: in a small-radius graph we are able either to satisfy all constraints or to find a small unsatisfiable sub-instance.

Lemma 8 *There is a polynomial time algorithm that given a linear unique game* $(G = (V,E), \{f_e\}_{e\in E}, S)$ *such that the graph G has radius at most t, and given a rooted spanning tree of G of depth at most t, either finds an assignment that satisfies all constraints or finds an unsatisfiable sub-instance of size at most* $4t+2$.

Proof: We try to construct an assignment A that satisfies all edges.

Let the value $x = A(r)$ of the root of the spanning tree be a "free variable," and then label every vertex u in G by a linear permutation ℓ_u of x consistent with the constraints along the edges of the spanning tree in the path from r to u.

Now look at all the edges of G that are not in the spanning tree. Each edge (u,v) determines a constraint of the form $\ell_u(x) = f_{u,v}(\ell_v(x))$, that is, $x = \ell_u^{(-1)}(f_{u,v}(\ell_v(x)))$, which is a linear equation of the form $ax+b = x$ with $a \neq 0$. Such an equation either has no solution, or it has exactly one solution, or it is satisfied for all x.

If there is an unsatisfied equation $\ell_u(x) = f_{u,v}(\ell_v(x))$, then we have a collection of at most $2t+1$ inconsistent constraints: the ones in the path from r to v in the spanning tree, plus the ones in the path from from r to u in the spanning tree, plus (u,v).

If there are two equations such that each one has only one solution and the two solutions are different, then we have a collection of at most $4t+2$ constraints that are inconsistent.

If every equation has at least one solution, and every one-solution equation (if any) is consistent with every other one, then clearly it is possible to satisfy all constraints. $\square$

We remark that we used the assumption of linearity in the above proof as follows: (i) the composition of several linear permutations is also a linear permutation, and (ii) a linear permutation has either zero, one or $|S|$ fixed points x such that $x = f(x)$. If, more generally, Π is a group of permutations such that, except for the identity, every permutation in Π has at most one fixed point, then unique games with permutations chosen from Π can be approximated as well as linear unique games.[5] The running time would be polynomial in n and in the time that it takes to compose permutations from Π and to check if a permutation is the identity.

We can now prove Theorem 2.

PROOF: [Of Theorem 2] Given a linear unique game $(G = (V,E), \{f_e\}, S)$ and a parameter ε we delete at most $\varepsilon|E|/2$ edges so that in the residual graph every connected component has radius at most $k = O(\varepsilon^{-1} \log |E|)$. We apply Lemma 8, and we either find an assignment that satisfies all the constraints in E' or we find a small "counterexample" of size at most $4k+2$.

In the former case we simply halt and output the assignment. In the latter case we remove the constraints in the unsatisfiable subinstance from G, and then recompute the low-radius decomposition, and so on.

Suppose that there is an assignment that satisfies at least a $1 - \varepsilon/(8t+4)$ fraction of constraints. Then, if we let U be the number of unsatisfiable sub-instances found by the algorithm, we must have $U \leq \varepsilon|E|/(8k+4)$, and so the number of edges removed through all phases is at most $\varepsilon|E|/2$. In the last phase, the algorithm removes at most $\varepsilon|E|/2$ other edges for the low-radius decomposition, and then satisfies all $(1-\varepsilon)|E|$ remaining constraints. $\square$

5 To be more precise, we should talk about an ensemble $\{\Pi_S\}_S$ of groups of permutations, one for each size of S.

5. d-to-d Games

This section follows once more the pattern of providing a good approximation in "nice," those with a low-radius constraint graph, and then giving a reduction from the general case to the nice case, using the Leighton-Rao decomposition theorem.

5.1. Approximation Algorithm for Low-Radius Instances

Lemma 9 *There is a polynomial time that, on input a satisfiable d-to-d game $(G = (V,E), \{P_e\}, S)$ such that G has radius k, finds an assignment that satisfies at least a $1/d^{2k}$ fraction of constraints.*

PROOF: Let r be a vertex of G such that every other vertex is at distance $\leq k$ from r, and let T be a spanning tree of G of depth k rooted at r. Let A be a satisfying assignment. "Guess" the value $A(r)$, then, for $i = 1, \ldots, k$, consider all vertices u at distance i from r and compute a list L_u of $\leq d^i$ values of S such that $A(u)$ is guaranteed to be an element of L_u. To do so, note that if we have computed a list L_u of size ℓ for a vertex u, and if (u,v) is an edge, then we can compute the list for v of size at most $d\ell$ by including, for every $a \in L_u$, the at most d values b such that $P_{u,v}(a,b) = 1$.

Once the lists have been computed, for every vertex u we randomly pick an assignment from L_u. Every vertex has probability at least $1/d^k$ of being assigned the correct value, and so every constraint has probability at least $1/d^{2k}$ of being satisfied. $\square$

5.2. Proof of Theorem 3

We now prove Theorem 3. Fix $t = 2^{\sqrt{(\log n)(\log d)}}$. Find a partition of the set of vertices of G such that each component has radius $k = O(\sqrt{(\log n)/(\log d)})$ and a $1/t$ fraction of the total weight of the edges are within the components. Lemma 9 gives us a way to satisfy a $1/d^{2k} = 1/2^{O(\sqrt{(\log n)(\log d)})}$ fraction of the edges within the component. Overall, we satisfy a $1/2^{O(\sqrt{(\log n)(\log d)})}$ fraction of constraints.

Acknowledgements

I wish to acknowledge Elchanan Mossel's contribution to this research, and his several very useful suggestions. I wish to thank Robi Krautghamer and James Lee for suggesting the use of semidefinite programming, proposing the SDP formulation (1) and helping with the proof of Claim 6.

I thank Kenji Obata for sharing his understanding of graph decomposition procedures while he was working on [Oba04].

References

[CKK$^+$05] Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D.Sivakumar. On the hardness of approximating multicut and sparsest-cut. In *Proceedings of the 20th IEEE Conference on Computational Complexity*, 2005.

[CMM05] Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Personal Communication, 2005.

[DMR05] Irit Dinur, Elchanan Mossel, and Oded Regev. Conditional hardness for approximate coloring. cs.CC/0504062, 2005.

[FL92] Uri Feige and László Lovász. Two-prover one round proof systems: Their power and their problems. In *Proceedings of the 24th ACM Symposium on Theory of Computing*, pages 733–741, 1992.

[FR04] Uriel Feige and Daniel Reichman. On systems of linear equations with two variables per equation. In *Proc. of APPROX-RANDOM'04*, pages 117–127, 2004.

[Kho02] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 767–775, 2002.

[KKMO04] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other two-variable CSPs? In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 146–154, 2004.

[KR03] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2-\varepsilon$. In *Proceedings of the 18th IEEE Conference on Computational Complexity*, 2003.

[KV05] Subhash Khot and Nisheeth Vishnoi. The unique games conjecture, integrality gap for cut problems and the embeddability of negative type metrics into ℓ_1. Manuscript, 2005.

[LR99] Frank T. Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46:787–832, 1999.

[MOO05] Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. math.PR/0503503, 2005.

[Oba04] Kenji Obata. Approximate max-integral-flow/min-multicut theorems. In *Proceedings of the 36th ACM Symposium on Theory of Computing*, pages 539–545, 2004.

[Raz98] Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998. Preliminary version in *Proc. of STOC'95*.

[Tre05] Luca Trevisan. Approximation algorithms for unique games. Technical Report TR05-34, Electronic Colloquium on Computational Complexity, 2005.