THEORY OF COMPUTING, Volume 4 (2008), pp. 111–128
http://theoryofcomputing.org

Approximation Algorithms for Unique Games*

Luca Trevisan†

Received: May 11, 2008; published: October 10, 2008.

Abstract: A *unique game* is a type of constraint satisfaction problem with two variables per constraint. The *value* of a unique game is the fraction of the constraints satisfied by an optimal solution. Khot (STOC'02) conjectured that for arbitrarily small $\gamma, \varepsilon > 0$ it is NP-hard to distinguish games of value smaller than γ from games of value larger than $1-\varepsilon$. Several recent inapproximability results rely on Khot's conjecture.

Considering the case of sub-constant ε, Khot (STOC'02) analyzes an algorithm based on semidefinite programming that satisfies a constant fraction of the constraints in unique games of value $1-O(k^{-10}\cdot(\log k)^{-5})$, where k is the size of the domain of the variables.

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 the constraints, where n is the number of variables. This is an improvement over Khot's algorithm if the domain is sufficiently large.

*An extended abstract of this paper appeared in the Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), pages 197–205.

†This material is based upon work supported by the National Science Foundation under grant No. CCF 0515231 and by the BSF under grant 2002246.

ACM Classification: F.2.2

AMS Classification: 68Q17

Key words and phrases: complexity theory, approximation algorithms, constraint satisfaction, unique games

DOI: 10.4086/toc.2008.v004a005

We also present a simpler algorithm for the special case of unique games with *linear* constraints, and a simple approximation algorithm for the more general class of *2-to-1* games.

1 Introduction

A *unique game* [8] is described by a set V of variables, taking values over a finite set S (we will refer to S as the *alphabet* of the games), 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 the 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 $S = \{a, b, c\}$:

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

Here we used the notation $\binom{a\ b\ c}{x\ y\ z}$ to represent the permutation f such that $f(a) = x$, $f(b) = y$ and $f(c) = z$. 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 is 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 $v = f_{v,u}(u)$. We also define the permutation $f_{u,v}$, $v > u$, as $f_{u,v} := f_{v,u}^{-1}$.

Given a unique game, if there exists an assignment that satisfies all constraints then it is easy to find such an assignment, as follows. 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. Fix a satisfying assignment A for the unique game; guess the value $A(r)$, 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.

If one is given an *almost* satisfiable unique game, that is, a game admitting an assignment that satisfies a $1-\gamma$ fraction of the constraints for some small γ, then no efficient algorithm is known to find a good assignment, that is, say, an assignment that satisfies a constant fraction of constraint independent of γ. Khot's [11] Unique Games Conjecture is that for every $0 < \gamma < 1/2$ there is a constant $c = c(\gamma)$ such that it is NP-hard to distinguish games of value $\geq 1-\gamma$ from games of value $\leq \gamma$, even when restricted to alphabets of size c and to graphs that are bipartite.[1]

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 factor [11], that the Vertex Cover problem is hard to approximate within $2-\varepsilon$ for every $\varepsilon > 0$ [14], that the Max Cut problem is hard to approximate within $.878\cdots$ [12, 18], and that the Sparsest Cut problem [5] is hard to approximate within any constant. Extensions of the Unique Games Conjecture to subconstant γ have been used to prove that the Sparsest Cut problem is hard to approximate within $\Omega((\log\log n)^{1/6})$ [15]. Several recent papers [23, 13, 2, 3, 20, 21, 17], too many to describe individually, have established inapproximability results for constraint satisfaction problems and other optimization problems by relying on the Unique Games Conjecture.

In partial support of this conjecture, Feige and Reichman [9] prove that it is NP-hard (under quasi-polynomial time reductions) to approximate the value of a unique game within a factor $2^{(\log n)^{.99}}$. 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, including [11, 14, 12, 18, 15, 5, 23, 13, 2, 3, 20, 21, 17], based on the Unique Games Conjecture, there is a good motivation to investigate algorithms that might contradict it. Prior to our work, the main algorithmic result on unique games was due to Khot [11], who presents an algorithm that given a unique game with an alphabet of size k and the promise that a $1-O(\varepsilon^5/k^{10}(\log k)^5)$ fraction of the constraints is satisfiable, finds an assignment that satisfies a $1-\varepsilon$ fraction of the constraints.

Khot [11] 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 the constraints satisfied by an optimal assignment. We will restrict ourselves to the case $d = d'$, which is

[1] Unique games on general graphs can be reduced to unique games on bipartite graphs, so the restriction to bipartite graphs is done with no loss of generality.

done without loss of generality. 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.[2]

An example of a 2-to-2 game is the 3-coloring problem. If G is a graph, consider the 2-to-2 game $(G=(V,E),\{\mathrm{neq}_e\}_{e\in E},\{a,b,c\})$, where the predicate $\mathrm{neq}_e(x,y)$ 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 [11] 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 [7]; as far as we know, there was no previous algorithmic investigation on its validity.

A preliminary version of this article appeared in FOCS 2005. The version that appeared in FOCS included an additional algorithm not described here; in this paper we provide the full proofs involved in the analysis of the other algorithms.

1.1 Our results

In this paper we rule out a generalization of the Unique Games Conjecture to the case of subconstant γ that could have been considered plausible given the result of Feige and Reichman [9].

Theorem 1.1. *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 there is an assignment that satisfies a*

$$1-\frac{c\cdot\varepsilon^3}{\log|E|}$$

fraction of the constraints, returns an assignment that satisfies at least a $1-\varepsilon$ fraction of the constraints. Furthermore, there is an algorithm that given the promise that the input unique game has value at least $1-\varepsilon$, finds an assignment that satisfies at least a $1/n^{O(\varepsilon)}$ fraction of the constraints. The algorithms run in time polynomial in $|E|$ and $|S|$.

The algorithm of Theorem 1.1, described in Section 3, works by solving and then rounding a semidefinite programming (SDP) relaxation of the unique game.

The restricted class of *linear* unique games has also been considered in the literature. Examples of linear unique games are those in which every constraint is a linear equation over a finite field, or an equation of the form $x=y+a$ where the variables x,y range over a finite group. More generally, we call a unique game *linear* if the permutations occurring in the instance generate a group of permutations in which every permutation other than the identity has at most one fixed point.

We have a stronger result, derived from a simpler algorithm, for linear unique games.

[2]The definition of 2-to-1 game given in [11] 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.

Theorem 1.2. *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 there is an assignment that satisfies a*

$$1 - \frac{c \cdot \varepsilon^2}{\log|E|}$$

fraction of the constraints, returns an assignment that satisfies at least a $1-\varepsilon$ fraction of the constraints. Furthermore, there is a constant c' and an algorithm that, on input a linear unique game $(G = (V,E),\{f_e\},S)$, and the promise that there is an assignment that satisfies a

$$1 - \frac{c' \cdot (\log\log|E|)}{\log|E|}$$

fraction of the constraints, returns an assignment that satisfies at least a

$$\frac{c' \cdot (\log\log|E|)}{\log|E|}$$

fraction of the constraints. The algorithms work in time polynomial in $|E|$ and (assuming an efficient representation of the group of permutations) $\log|S|$.

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 [22], for which it is NP-hard (under quasi-polynomial time reductions) to distinguish satisfiable instances from instances of value smaller than $1/2^{(\log n)^{.99}}$. Our algorithm also works for d-to-d games for large (even super-constant) d.

Theorem 1.3. *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 the constraints. The algorithm runs in time polynomial in $|E|$ and $|S|$.

In particular, Theorem 1.3 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 the 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 diameter; and (ii) we show how to reduce general instances to nice ones, via a result of Leighton and Rao [16] that gives a decomposition of general graphs into low-diameter components.

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 instances where the constraint graph has small diameter and where every edge gives a large contribution to the objective function. We then give a reduction from the general case to such instances.

Our main result is a rounding algorithm with the following guarantee: given a solution for the SDP such that every edge contributes at least $1-\delta$ to the objective function, and assuming that the constraint graph has diameter d, we can find in polynomial time a solution that satisfies at least a $1-O(d\delta)$ fraction of the 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 the edges. Then we use a decomposition procedure by Leigthon and Rao that shows how to decompose a graph in components of diameter $d = O(\varepsilon^{-1}\log n)$ by removing only an ε fraction of the 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 the edges in each component. Overall, we have satisfied at least a $1-O(\varepsilon)$ fraction of the edges.

Algorithm for linear unique games

In the case of linear unique games, our main (and simple) technical lemma is that if the constraint graph has diameter d, then we can either find an assignment that satisfies all edges, or we can find a set of at most $4d+2$ constraints that cannot be simultaneously satisfied.

With this preliminary result in place, our algorithm works as follows: first, using the algorithm of Leighton and Rao, it computes a decomposition of the graph such that, after removing at most an ε fraction of the edges, every connected component has diameter $O(\log n)$. Then we try to satisfy all the constraints of the residual graph, by applying our lemma to each connected component. Our lemma guarantees that either we satisfy all constraints within all components, or we find an inconsistent set of at most $O(\log n)$ constraints. In the former case, we stop; in the latter case we remove the $O(\log n)$ inconsistent constraints from the graph, we re-compute a Leighton-Rao decomposition, and continue.

Our analysis of the algorithm relies on the fact that if the algorithm runs through T phases before stopping, then it has found a collection of $O(T\log n)$ constraints such that every assignment contradicts at least T of them. If we started from the promise that there is an assignment that satisfies at least a $1-O(\varepsilon/\log n)$ fraction of the constraints, then the algorithm must halt within $T = O(\varepsilon\cdot|E|/\log n)$ steps, and hence it removes at most $O(\varepsilon\cdot|E|)$ constraints, before converging on a sub-instance that can be completely satisfied.

Algorithm for 2-to-1 games

For 2-to-1 games our main result is that if we have a satisfiable instance in which the constraint graph has diameter t, then it is possible to efficiently satisfy at least a $1/2^t$ fraction of the constraints.

Our algorithm considers a spanning tree of diameter t of the constraint graph, guesses the value $A(r)$ of a satisfying assignment at the root, and then, for every vertex u, compiles a list $L_u \subseteq S$ of values that are compatible with r having value $A(r)$. For a vertex at distance i from r, the size of the list is at most 2^i, and so for every vertex u the size of L_u is at most 2^t. The algorithm then, for every vertex u, picks at random an assignment from L_u: we see that every constraint is satisfied with probability at least $1/2^{2t}$, and so on average at least a $1/2^{2t}$ fraction of the constraints is satisfied.

To apply our algorithm to general instances, we first use the Leighton-Rao algorithm to find a low diameter decomposition of the constraint graph such that each component has diameter at most $t = O(\sqrt{\log n})$. This can be done by removing at most a $1-1/2^{O(\sqrt{\log n})}$ fraction of the edges.

Having started from a satisfiable instance, each component in the decomposition is still satisfiable, and so our algorithm will satisfy at least a $1/2^{2t} = 1/2^{O(\sqrt{\log n})}$ fraction of the constraints in each connected component.

If we have a d-to-d game, a generalization of the algorithm described above can satisfy a $1/d^{2t}$ fraction of the constraints in an instance of diameter t. In general instances, we first find a decomposition of diameter $k = O(\sqrt{\log n \cdot (\log d)^{-1}})$, which can be done after removing at most a $1-1/2^{\sqrt{\log n \cdot \log d}}$ fraction of the edges, and then our algorithm will satisfy at least a $1/2^{O(\sqrt{\log n \cdot \log d})}$ fraction of the constraints in each component.

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 of the quality of the approximation on $|E|$ could be replaced by analogous dependencies on $|V|$.

1.4 Later results

A number of new algorithmic results for unique games have been discovered since the initial announcement of our results.

Gupta and Talwar [10] present an algorithm that, given a unique game of value $1-\varepsilon$, satisfies a $1-O(\varepsilon \log n)$ fraction of the constraints, that is, they are able to approximate the problem of *minimizing the number of contradicted constraints* within a $O(\log n)$ factor. Their algorithm is based on linear programming.

Charikar, Makarychev, and Makarychev [4] devise an improved SDP-based approximation algorithm for unique games. Given a unique game of value $1-\varepsilon$, their algorithm satisfies a $1/|S|^{O(\varepsilon)}$ fraction of the constraints. In particular, their algorithm satisfies a constant fraction of the constraints in a unique game of value $1-O(1/\log |S|)$.

In an instance of value $1-\varepsilon$, our algorithm only satisfies a $1/n^{O(\varepsilon)}$ fraction of the constraints. In the interesting case $|S| = O(\log n)$, the approximation $1/|S|^{O(\varepsilon)}$ of the algorithm of Charikar et al. is exponentially better.

Chlamtac, Makarychev and Makarychev [6] present an alternative SDP-based approximation algorithm that satisfies a $1-O(\varepsilon\sqrt{\log n \cdot \log k})$ fraction of the constraints in a unique game of value $1-\varepsilon$.

Arora et al. [1] shows that if the constraint graph of a given unique game is an expander, then applying our rounding strategy to an SDP solution of cost $1-\varepsilon$ satisfies a $1-O(\varepsilon)$ fraction of the constraints. Their result contradicts the restriction of the Unique Games Conjecture to instances for which the constraint graph is expanding.

2 Preliminaries

2.1 The Leighton-Rao graph decomposition

We say that a graph $G = (V,E)$ has diameter at most d if every two vertices are connected by a path of length at most d.

The following result is due to Leighton and Rao [16] and is used in several places in this paper.

Lemma 2.1 (Leighton and Rao [16]). *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 diameter at most $2\cdot(1+\log|E|)/\log t$.*

We note that, in particular, if we want $|E'| \geq (1-\varepsilon)\cdot|E|$, the diameter of the connected components can be $O(\varepsilon^{-1}\cdot\log|E|)$.

2.2 The semidefinite programming relaxation of unique games

In a *semidefinite program* each variable is a vector, and the objective function and the constraints are linear in the inner products of the vectors.

To give a semidefinite programming relaxation of unique games, it is convenient to first give an *integer programming* formulation of the program, in which variables are constrained to be integers, and then relax the formulation to a semidefinite programming relaxation.

We consider the following integer programming formulation of unique games. We assume without loss of generality that the set S equals $\{1,\dots,|S|\}$. For every variable u of the unique game, we have $k := |S|$ boolean variables $u_1,\dots,u_k$ in the integer program, with the intended meaning that if $u = i$ then $u_i = 1$ and $u_j = 0$ for $j \neq i$. The objective function contains one term $\sum_{i\in S} v_{f(i)}u_i$ for each constraint $v = f(u)$. Hence, our integer programing formulation is as follows:

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

We can obtain a first semidefinite programming relaxation by replacing each integral variable u_i with a vector variable $\mathbf{u}_i$, and integer multiplication with inner products.

$$\begin{array}{ll} \max & \sum_{e=(u,v)\in E} \sum_{i\in S} \mathbf{v}_{f_e(i)} \mathbf{u}_i \\ \text{Subject to} & \\ \quad \mathbf{u}_i \cdot \mathbf{u}_j = 0 & (\forall u \in V, \forall i,j \in [k], i \neq j) \\ \quad \sum_{i\in S} \|\mathbf{u}_i\|^2 = 1 & (\forall u \in V) \end{array} \tag{2.1}$$

We find it convenient to formulate the objective function in a different, but equivalent, way. Let $\{\mathbf{v}\}_{v\in V}$ be a feasible solution to (2.1) and let $v = f_e(u)$ be a constraint of the unique game. Then the contribution of the constraint to the objective function satisfies

$$\sum_{i\in S} \|\mathbf{v}_{f_e(i)} - \mathbf{u}_i\|^2 = \sum_{i\in S} \|\mathbf{v}_{f_e(i)}\|^2 + \|\mathbf{u}_i\|^2 - 2\mathbf{v}_{f_e(i)} \cdot \mathbf{u}_i = 2 - 2\sum_{i\in S} \mathbf{v}_{f_{u,v}(i)} \,.$$

This means that we can equivalently rewrite the objective function of (2.1) as

$$\sum_{e=(u,v)\in E} \left(1 - \sum_{i\in S} \frac{1}{2} \|\mathbf{v}_{f_e(i)} - \mathbf{u}_i\|^2 \right) .$$

The semidefinite programming solution that we shall use has the above formulation of the objective function and, in addition, the “triangle inequalities”:

$$\begin{array}{ll} \max & \sum_{e=(u,v)\in E} \left(1 - \sum_{i\in S} \frac{1}{2} \|\mathbf{v}_{f_e(i)} - \mathbf{u}_i\|^2\right) \\ \text{Subject to} & \\ \quad \mathbf{u}_i \cdot \mathbf{u}_j = 0 & (\forall u \in V, \forall i,j \in [k], i \neq j) \\ \quad \sum_{i\in S} \|\mathbf{u}_i\|^2 = 1 & (\forall u \in V) \\ \quad \|\mathbf{w}_h - \mathbf{u}_i\|^2 \leq \|\mathbf{w}_h - \mathbf{v}_j\|^2 + \|\mathbf{v}_j - \mathbf{u}_i\|^2 & (\forall u,v,w \in V, \forall i,j,h \in [k]) \\ \quad \|\mathbf{v}_j - \mathbf{u}_i\|^2 \geq \|\mathbf{v}_j\|^2 - \|\mathbf{u}_i\|^2 & (\forall u,v \in V, \forall i,j \in [k]) \end{array} \tag{2.2}$$

The “triangle inequalities” are valid for integral 0/1 solutions, because if $a,b \in \{0,1\}$ then $|a-b|^2 = |a-b|$, and so they reduce to the standard triangle inequalities for the absolute value.

Feige and Lovász [8] and Khot [11] use a slightly different formulation, and it is not clear (to us) whether the SDP (2.2) is equivalent or not to the ones in [8, 11].

3 The SDP-based algorithm for general unique games

We shall first present a rounding scheme that works well in instances where the constraint graph has small diameter and where every edge gives a large contribution to the objective function; then we will use the Leighton-Rao decomposition theorem to reduce general instances to instances of logarithmic diameter.

3.1 The rounding procedure: Analysis on "nice" instances

In this section we shall analyze the following randomized rounding procedure for solutions to the SDP (2.2).

Algorithm ROUND
Input: a vector solution $\{\mathbf{v}_i\}_{v\in V, i\in[k]}$ to the SDP (2.2)
Output: an assignment $A : V \to [k]$ to the variables of the unique game

- Fix arbitrarily a vertex r
- Pick at random a value $i \in [k]$ with probability $\|\mathbf{r}_i\|^2$, and set $A(r) := i$;
- For every other variable v, find the j that minimizes the "distance" $\|\mathbf{v}_j - \mathbf{r}_i\|^2$, and assign $A(v) := j$.

We shall prove that algorithm ROUND satisfies a large fraction of the constraints, on average, provided that the constraint graph has small diameter and that we start from a vector solution in which every edge has a large contribution to the objective function.

Lemma 3.1. *Let $(G, [k], \{f_e\})$ be a unique game such that G has diameter d, and let $\{\mathbf{v}_i\}_{v\in V, i\in[k]}$ be a vector solution to the SDP (2.2) such that every edge contributes at least $1-\delta$ to the objective function. Then, for every constraint $v = f_{v,u}(u)$ in the unique game, the probability that the assignment of algorithm ROUND satisfies the constraint is at least $1 - \delta \cdot (8d+4)$.*

Before going into the proof of Lemma 3.1, it is helpful to first consider the simplified case in which all the constraints in the unique game are equality constraints. (Indeed, this case contains all the difficulties of the general case.) The assumption that each constraint contributes at least $1-\delta$ to the objective function means that for every vertices u, v for which we have the constraint $u = v$ we also have

$$\sum_{i\in[k]} \|\mathbf{u}_i - \mathbf{v}_i\|^2 \le 2\delta \,.$$

In general, we may think of our vector solution as defining a "distance" $\sum_{i\in[k]} \|\mathbf{u}_i - \mathbf{v}_i\|^2$ between any two pairs of vertices u, v, and, because of the triangle inequalities in SDP (2.2), this "distance" satisfies the triangle inequality as well. Since any two vertices are within distance at most d from each other (and, in particular, from r) in the constraint graph, we have

$$\sum_{i\in[k]} \|\mathbf{r}_i - \mathbf{v}_i\|^2 \le 2d\delta$$

for every vertex v. A geometric interpretation of algorithm ROUND (Lemma 3.2 below) allows us to conclude that $A(r) = A(v)$ with probability at least $1 - 4d\delta$. This implies that for every constraint $u = v$ there is a probability at least $1 - 8d\delta$ that we have both $A(r) = A(u)$ and $A(r) = A(v)$, and so the constraint is satisfied with probability at least $1 - 8d\delta$.

Turning to the proof of Lemma 3.1, we begin with the following result about the distribution of outputs of algorithm ROUND.

Lemma 3.2. *Let $(G,[k],\{f_e\})$ be a unique game, $\{\mathbf{v}_i\}_{v\in V, i\in[k]}$ be a vector solution to the SDP (2.2), v be a vertex and f be a permutation such that*

$$\sum_{i\in[k]} \|\mathbf{r}_i - \mathbf{v}_{f(i)}\|^2 \leq \delta\,.$$

Then, with probability at least $1-2\delta$ over the random choices of algorithm ROUND we have $A(v) = f(A(r))$.

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

$$\mathbb{P}[A(v) \neq f(A(r))] = \sum_{i\in B} \|\mathbf{r}_i\|^2\,.$$

Now we claim that, for every $i \in B$, we have

$$\|\mathbf{r}_i\|^2 \leq 2\cdot\|\mathbf{r}_i - \mathbf{v}_{f(i)}\|^2\,.$$

This follows from the following geometric claim, by setting $\mathbf{c} \leftarrow \mathbf{r}_i$, $\mathbf{b} \leftarrow \mathbf{v}_j$, $\mathbf{a} \leftarrow \mathbf{v}_{f(i)}$, where j is the minimizer of $\|\mathbf{r}_i - \mathbf{v}_j\|^2$.

Claim 3.3. *Let $\mathbf{a},\mathbf{b},\mathbf{c}$ be three vectors such that $\{\mathbf{0},\mathbf{a},\mathbf{b},\mathbf{c}\}$ is a metric space with respect to the Euclidean-squared distance, $\mathbf{a},\mathbf{b}$ are orthogonal, and $\|\mathbf{c}-\mathbf{a}\|^2 \geq \|\mathbf{c}-\mathbf{b}\|^2$. Then*

$$\|\mathbf{c}\|^2 \leq 2\cdot\|\mathbf{c}-\mathbf{a}\|^2\,.$$

Proof. We consider three cases: if $\mathbf{c}$ is a long vector relative to $\mathbf{a}$, then the claim is immediate; if $\mathbf{c}$ is short relative to $\mathbf{a}$, but long relative to $\mathbf{b}$, then it must be far from $\mathbf{b}$ and, for a stronger reason, from $\mathbf{a}$. Finally, if $\mathbf{c}$ is about the same length as $\mathbf{b}$ and $\mathbf{a}$, then, considering that $\mathbf{b}$ and $\mathbf{a}$ form a 90 degree angle, and that $\mathbf{c}$ is closer to $\mathbf{b}$ than to $\mathbf{a}$, it must be quite far from $\mathbf{a}$.

Formally, the three cases are:

1. If $\|\mathbf{a}\|^2 \leq \frac{1}{2}\|\mathbf{c}\|^2$, then $\|\mathbf{c}-\mathbf{a}\|^2 \geq \|\mathbf{c}\|^2 - \|\mathbf{a}\|^2 \geq \frac{1}{2}\|\mathbf{c}\|^2$.

2. If $\|\mathbf{b}\|^2 \leq \frac{1}{2}\|\mathbf{c}\|^2$, then $\|\mathbf{c}-\mathbf{a}\|^2 \geq \|\mathbf{c}-\mathbf{b}\|^2 \geq \|\mathbf{c}\|^2 - \|\mathbf{b}\|^2 \geq \frac{1}{2}\|\mathbf{c}\|^2$.

3. If $\|\mathbf{a}\|^2, \|\mathbf{b}\|^2 \geq \frac{1}{2}\|\mathbf{c}\|^2$, then from the triangle inequality

$$\|\mathbf{b}-\mathbf{a}\|^2 \leq \|\mathbf{b}-\mathbf{c}\|^2 + \|\mathbf{c}-\mathbf{a}\|^2 \leq 2\|\mathbf{c}-\mathbf{a}\|^2$$

and by the Pythagorean theorem and the orthogonality of $\mathbf{a}$ and $\mathbf{b}$ we have

$$\|\mathbf{b}-\mathbf{a}\|^2 = \|\mathbf{b}\|^2 + \|\mathbf{a}\|^2$$

so that

$$\|\mathbf{c}-\mathbf{a}\|^2 \geq \frac{1}{2}\|\mathbf{b}-\mathbf{a}\|^2 = \frac{1}{2}\|\mathbf{a}\|^2 + \frac{1}{2}\|\mathbf{b}\|^2 \geq \frac{1}{2}\|\mathbf{c}\|^2\,.$$

□

Hence,

$$\mathbb{P}[A(v) \neq f(A(r))] = \sum_{i \in B} \|\mathbf{r}_i\|^2 \leq 2 \sum_{i \in B} \|\mathbf{r}_i - \mathbf{v}_{f(i)}\|^2 \leq 2 \sum_{i \in [k]} \|\mathbf{r}_i - \mathbf{v}_{f(i)}\|^2 \leq 2\delta \,.$$

□

We can now analyze how algorithm ROUND performs on any given constraint and prove Lemma 3.1.

Proof of Lemma 3.1. Let (u,v) be a constraint in G, and consider the path $r = u^0, u^1, \ldots, u^t = u, u^{t+1} = v$ where $r, u^1, \ldots, u^{t-1}, u$ is a path from r to u of length $t \leq d$ in G.

For $\ell = 0, \ldots, t+1$, let f_i be the composition of the permutations corresponding to the path from r to u^ℓ, that is

$$f_0(x) := x$$

$$f_\ell(x) := f_{(u^\ell, u^{\ell-1})}(f_{\ell-1}(x)) \,.$$

We claim that there is a probability at least $1 - \delta \cdot 4t$ that $A(u) = f_t(A(r))$.

By writing the difference $\mathbf{r}_i - \mathbf{u}_{f_t(i)}$ as a telescopic sum, and then using the triangle inequality in the SDP formulation, we have

$$\begin{aligned}
\sum_{i \in [k]} \|\mathbf{r}_i - \mathbf{u}_{f_t(i)}\|^2 &= \sum_{i \in [k]} \|\sum_{\ell=0}^{t-1} \mathbf{u}^\ell_{f_\ell(i)} - \mathbf{u}^{\ell+1}_{f_{\ell+1}(i)}\|^2 \\
&\leq \sum_{i \in [k]} \sum_{\ell=0}^{t-1} \|\mathbf{u}^\ell_{f_\ell(i)} - \mathbf{u}^{\ell+1}_{f_{\ell+1}(i)}\|^2 \\
&= \sum_{\ell=0}^{t-1} \sum_{j \in [k]} \|\mathbf{u}^\ell_j - \mathbf{u}^{\ell+1}_{f_{u^\ell, u^{\ell+1}}(j)}\|^2 \\
&\leq t \cdot 2\delta \,.
\end{aligned}$$

Now we can invoke Lemma 3.2 and see that the probability that $A(u) = f_t(A(r))$ is at least $1 - \delta \cdot 4t$.

The same argument shows that with probability at least $1 - \delta \cdot 4 \cdot (t+1)$ we $A(v) = f_{t+1}(A(r))$. By a union bound, it follows that there is a probability at least $1 - \delta \cdot (8t+4)$ that both events happen, in which case $A(v) = f_{(v,u)}(A(u))$. □

3.2 The general case: Proof of Theorem 1.1

In this section we prove the following result.

Theorem 3.4. *There is a constant c and a polynomial time algorithm, that, given an instance of unique games and a solution to the SDP (2.2) of cost at least*

$$\left(1 - \frac{c\varepsilon^3}{\log n}\right) \cdot |E| \,,$$

finds a solution for the unique game that satisfies at least a $1 - \varepsilon$ fraction of the constraints.

Suppose that the SDP relaxation of a unique game $(G=(V,E),S,\{f_e\}_{e\in E})$ 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 the constraints, their contribution to the objective function is at least $1-3\gamma/\varepsilon=1-3c\varepsilon^2/\log n$.

The algorithm of Theorem 3.4 is as follows:

1. Remove the constraints whose contribution is smaller than $1-3\gamma/\varepsilon$.
2. Apply the Leighton-Rao decomposition of Lemma 2.1 with $t=1/(1-\varepsilon/3)$ to the residual graph.
3. Use the algorithm of Lemma 3.1 to satisfy at least $1-\varepsilon/3$ fraction of the constraints in each connected component of the residual graph that we obtain after steps (1) and (2).

The first 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.

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

The third step can be executed as required, provided that the constant c satisfies

$$1-\frac{3c\varepsilon^2}{(\log n)}\geq 1-\frac{\varepsilon}{24(d+1)},$$

that is, $c<(\log n)/(72\cdot(d+1)\cdot\varepsilon)$. Since we had $d=O((\log n)/\varepsilon)$, c is indeed an absolute constant.

Overall, we have found a solution that contradicts at most $\varepsilon\cdot|E|$ constraints, and so we have a proof of Theorem 3.4 and of the first part of Theorem 1.1.

To prove the "furthermore" part of Theorem 1.1, we use the same algorithm but set the parameters as follows: in the first step, remove the edges that contribute less than $1-\delta/2$ to the objective function, and in the second step, find in the residual graph a decomposition of diameter, say, $1/(100\delta)$ in which at least a $1/n^{O(\delta)}$ fraction of the edges are not removed.

4 Linear unique games

We call a group of permutations *linear* if every permutation other than the identity has at most one fixed point, and we call an instance of unique games a *linear unique game* if the permutations occurring in the instance generate a linear group.

Linear unique games arise in contexts in which every constraint is an algebraic equation. One example of linear unique games are those where $S=\mathbb{F}$, for a finite field $\mathbb{F}$, and every constraint is a linear equation $x=ay+b$, $a\neq 0$. Another example is given by games where $S=\mathbb{F}$ and every constraint is an equation of the form $x-y=a$.

As before, we first prove a result that applies only to instances with a small-diameter constraint graph, and then use the Leighton-Rao decomposition to extend the algorithm to arbitrary instances.

4.1 The low-diameter case

If we are given a linear unique game in a small-diameter graph, we are able either to satisfy all constraints or to find a small unsatisfiable sub-instance.

Lemma 4.1. *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 diameter at most d, and given a rooted spanning tree T of G of depth at most d, either finds an assignment that satisfies all constraints or finds an unsatisfiable sub-instance of size at most* $4d+2$*.*

Proof. For every vertex v, denote by ℓ_v the composition of the permutations f_e along the path from r to v in T; let ℓ_r be the identity permutation.

Our algorithm considers, for each $i \in S$, the assignment $A_i(v) := \ell_v(i)$. Each such assignment clearly satisfies all the constraints in the spanning tree.

Consider a constraint $v = f_{v,u}(u)$ not in the spanning tree. Then the constraint is satisfied by A_i provided that $l_v(i) = f_{v,u}\ell_u(i)$, that is, $i = l_v^{-1}(f_{v,u}(\ell_u(i)))$. This means that A_i satisfies the constraint if and only if i is a fixed point of the linear permutation $l_v^{-1}(f_{v,u}(\ell_u(\cdot)))$. There are three possibilities:

1. The permutation $l_v^{-1}(f_{v,u}(\ell_u(\cdot)))$ is the identity; then the constraint will be satisfied by every A_i.

2. The permutation $l_v^{-1}(f_{v,u}(\ell_u(\cdot)))$ has exactly one fixed point $i_{u,v}$; we call it the *critical value* for the constraint (u,v).

3. The permutation $l_v^{-1}(f_{v,u}(\ell_u(\cdot)))$ has no fixed point. then we say that (u,v) is *inconsistent*.

If the unique game contains an inconsistent constraint (u,v), then the path from r to u in T, the path from r to v in T, and the constraint (u,v) form an unsatisfiable subinstance of size $2d+1$.

If the unique game contains two constraints (u,v) and (w,z) having two distinct critical values, then the paths from r to u, v, w and z in T, plus the two constraints (u,v) and (w,z) are an unsatisfiable sub-instance of size $4d+2$.

In all other cases, there is an i such that A_i satisfies all the constraints. ☐

4.2 The general case

We can now prove Theorem 1.2.

Proof of Theorem 1.2. Given a linear unique game $(G = (V,E), \{f_e\}, S)$ and a parameter ε we apply the Leighton-Rao decomposition and delete at most $\varepsilon|E|/2$ edges so that in the residual graph every connected component has diameter at most $d = O(\varepsilon^{-1} \log |E|)$. We apply Lemma 4.1 in each connected component, and we either find an assignment that satisfies all the constraints in E' or we find a small "counterexample" of size at most $4d+2$.

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-diameter decomposition, and so on. Formally, the algorithm is as follows

1. Apply the Leighton-Rao algorithm of Lemma 2.1 to G, and remove at most $\varepsilon|E|/2$ edges from G so that in the residual graph G' every connected component has diameter d.

2. Apply the algorithm of Lemma 4.1 to each connected component of G'.

3. If the previous step finds satisfying assignments that, together, satisfy all the constraints of G', output the assignments and halt.

4. Otherwise, remove from G the "counterexamples" of size $\leq 4d+2$ found in step (2) and go to step (1).

Suppose that there is an assignment that satisfies at least a $1-\varepsilon/(8d+4)$ fraction of the constraints. Then, if we let U be the number of unsatisfiable sub-instances found by the algorithm, we must have $U \leq \varepsilon|E|/(8d+4)$, 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-diameter decomposition, and then satisfies all $(1-\varepsilon)|E|$ remaining constraints.

To prove the "furthermore" part of Theorem 1.2, we use decompositions into components of diameter $2\cdot(\log 2|E|)/\log\log|E|$, which can be found by deleting at most a $1/\sqrt{\log|E|}$ fraction of the edges, and then use the same analysis. $\square$

5 d-to-d games

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

5.1 Approximation algorithm for low-diameter instances

Lemma 5.1. *There is a polynomial time algorithm that, on input a satisfiable d-to-d game $(G=(V,E),\{P_e\},S)$ such that G has diameter t, finds an assignment that satisfies at least a $1/d^{2k}$ fraction of the constraints.*

Proof. Let r be a vertex of G such that every other vertex is at distance $\leq t$ from r, and let T be a spanning tree of G of depth t rooted at r. Let A be a satisfying assignment. "Guess" the value $A(r)$, then, for $i=1,\dots,t$, 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 $1/d^t$ of being assigned the correct value, and so every constraint has probability at least $1/d^{2t}$ of being satisfied. $\square$

5.2 Proof of Theorem 1.3

We now prove Theorem 1.3. Fix $t=2^{\sqrt{\log n\cdot\log d}}$. Find a partition of the set of vertices of G such that each component has diameter $k=O(\sqrt{\log n/\log d})$ and a $1/t$ fraction of the total weight of the edges are within the components. Lemma 5.1 gives us a way to satisfy a $1/d^{2k}=1/2^{O(\sqrt{\log n\cdot\log d})}$ fraction of the edges within the component. Overall, we satisfy a $1/2^{O(\sqrt{\log n\cdot\log d})}$ fraction of the constraints.

Acknowledgements

I wish to acknowledge Elchanan Mossel's contribution to this research, and his several very useful suggestions. I wish to thank Robi Krauthgamer and James Lee for suggesting the use of semidefinite programming, proposing the SDP formulation (2.2) and helping with the proof of Claim 3.3.

I thank Kenji Obata for sharing his understanding of graph decomposition procedures while he was working on [19]. Avi Eyal, Piotr Indyk, Salil Vadhan, and the anonymous referees, among others, gave very helpful feedback about the presentation of our algorithms.

References

[1] * SANJEEV ARORA, SUBHASH KHOT, ALEXANDRA KOLLA, DAVID STEURER, MADHUR TULSIANI, AND NISHEETH VISHNOI: Unique games on expanding constraint graphs are easy. In *Proc. 40th STOC*, pp. 21–28. ACM Press, 2008. [STOC:10.1145/1374376.1374380]. 1.4

[2] * PER AUSTRIN: Balanced max 2-SAT might not be the hardest. In *Proc. 39th STOC*, pp. 189–197. ACM Press, 2007. [STOC:10.1145/1250790.1250818]. 1

[3] * PER AUSTRIN: Towards sharp inapproximability for any 2-CSP. In *Proc. 48th FOCS*, pp. 307–317. IEEE Computer Society, 2007. [FOCS:10.1109/FOCS.2007.73]. 1

[4] * MOSES CHARIKAR, KONSTANTIN MAKARYCHEV, AND YURY MAKARYCHEV: Near-optimal algorithms for unique games. In *Proc. 38th STOC*, pp. 205–214. ACM Press, 2006. [STOC:1132516.1132547]. 1.4

[5] * SHUCHI CHAWLA, ROBERT KRAUTHGAMER, RAVI KUMAR, YUVAL RABANI, AND D.SIVAKUMAR: On the hardness of approximating multicut and sparsest-cut. In *Proc. 20th IEEE Conf. on Computational Complexity*, pp. 144–153. IEEE Computer Society, 2005. [CCC:10.1109/CCC.2005.20]. 1

[6] * EDEN CHLAMTAC, KONSTANTIN MAKARYCHEV, AND YURY MAKARYCHEV: How to play unique games using embeddings. In *Proc. 47th FOCS*, pp. 687–696. IEEE Computer Society, 2006. [FOCS:10.1109/FOCS.2006.36]. 1.4

[7] * IRIT DINUR, ELCHANAN MOSSEL, AND ODED REGEV: Conditional hardness for approximate coloring. In *Proc. 38th STOC*, pp. 344–353. ACM Press, 2006. [STOC:1132516.1132567]. 1

[8] * URI FEIGE AND LÁSZLÓ LOVÁSZ: Two-prover one round proof systems: Their power and their problems. In *Proc. 24th STOC*, pp. 733–741. ACM Press, 1992. [STOC:129712.129783]. 1, 2.2

[9] * URIEL FEIGE AND DANIEL REICHMAN: On systems of linear equations with two variables per equation. In *7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'04)*, pp. 117–127. Springer, 2004. [Springer:uk7fpuul45c5h6j5]. 1, 1.1

[10] * ANUPAM GUPTA AND KUNAL TALWAR: Approximating unique games. In *Proc. 17th ACM-SIAM Symp. on Discrete Algorithms (SODA'06)*, pp. 99–106. ACM Press, 2006. [SODA:1109557.1109569]. 1.4

[11] * SUBHASH KHOT: On the power of unique 2-prover 1-round games. In *Proc. 34th STOC*, pp. 767–775. ACM Press, 2002. [STOC:509907.510017]. 1, 2, 2.2

[12] * SUBHASH KHOT, GUY KINDLER, ELCHANAN MOSSEL, AND RYAN O'DONNELL: Optimal inapproximability results for MAX-CUT and other two-variable CSPs? In *Proc. 45th FOCS*, pp. 146–154. IEEE Computer Society, 2004. [FOCS:10.1109/FOCS.2004.49]. 1

[13] * SUBHASH KHOT AND RYAN O'DONNELL: SDP gaps and UGC-hardness for MAX-CUTGAIN. In *Proc. 47th FOCS*, pp. 217–226. IEEE Computer Society, 2006. [FOCS:10.1109/FOCS.2006.67]. 1

[14] * SUBHASH KHOT AND ODED REGEV: Vertex cover might be hard to approximate to within $2-\varepsilon$. *J. Comput. System Sci.*, 74:335–349, 2008. Preliminary version in *Proc. CCC'03*. [JCSS:10.1016/j.jcss.2007.06.019]. 1

[15] * SUBHASH KHOT AND NISHEETH VISHNOI: The unique games conjecture, integrality gap for cut problems and the embeddability of negative type metrics into ℓ_1. In *Proc. 46th FOCS*, pp. 53–63. IEEE Computer Society, 2005. [FOCS:10.1109/SFCS.2005.74]. 1

[16] * FRANK T. LEIGHTON AND SATISH RAO: Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46:787–832, 1999. [JACM:331524.331526]. 1.2, 2.1, 2.1

[17] * RAJESKAR MANOKARAN, SEFFI NAOR, PRASAD RAGHAVENDRA, AND ROY SCHWARTZ: SDP gaps and UGC hardness for multiway cut, 0-extension and metric labeling. In *Proc. 40th STOC*, pp. 11–20. ACM Press, 2008. [STOC:10.1145/1374376.1374379]. 1

[18] * ELCHANAN MOSSEL, RYAN O'DONNELL, AND KRZYSZTOF OLESZKIEWICZ: Noise stability of functions with low influences: invariance and optimality. In *Proc. 46th FOCS*, pp. 21–30. IEEE Computer Society, 2005. [FOCS:10.1109/SFCS.2005.53]. 1

[19] * KENJI OBATA: Approximate max-integral-flow/min-multicut theorems. In *Proc. 36th STOC*, pp. 539–545. ACM Press, 2004. [STOC:1007352.1007433]. 5.2

[20] * RYAN O'DONNELL AND YI WU: An optimal SDP algorithm for Max-Cut, and equally optimal long code tests. In *Proc. 40th STOC*, pp. 335–344. ACM Press, 2008. [STOC:10.1145/1374376.1374425]. 1

[21] * PRASAD RAGHAVENDRA: Optimal algorithms and inapproximability results for every CSP? In *Proc. 40th STOC*, pp. 245–254. ACM Press, 2008. [STOC:10.1145/1374376.1374414]. 1

[22] * RAN RAZ: A parallel repetition theorem. *SIAM J. Computing*, 27(3):763–803, 1998. Preliminary version in *STOC'95*. [SICOMP:10.1137/S0097539795280895, STOC:225058.225181]. 1.1

[23] * ALEX SAMORODNITSKY AND LUCA TREVISAN: Gowers uniformity, influence of variables, and PCPs. In *Proc. 38th STOC*, pp. 11–20. ACM Press, 2006. [STOC:10.1145/1132516.1132519]. 1

AUTHOR

Luca Trevisan
Associate Professor
Department of Electrical Engineering and Computer Science
University of California
Berkeley, CA 94720-1776
luca@EECS.Berkeley.edu
http://www.cs.berkeley.edu/~luca

ABOUT THE AUTHOR

LUCA TREVISAN is an associate professor of computer science at U. C. Berkeley. Luca received his Laurea (B. Sc.) degree in 1993 and his Dottorato (Ph. D.) in 1997, both from the University of Rome La Sapienza, advised by Pierluigi Crescenzi. Before coming to Berkeley in 2000, Luca was a post-doc at MIT and at DIMACS, and an assistant professor at Columbia University.

Luca's research is in theoretical computer science, and most of his work has been about the approximability of combinatorial optimization problems and about the power of randomness in computation.

Luca received the STOC'97 student paper award and the 2000 Oberwolfach Prize. He was an invited speaker at ICM 2006 in Madrid.

Luca lives, beyond his means, in expensive San Francisco, where he enjoys the film festivals, the neighborhood coffee shops, and the food. When not in San Francisco or Berkeley, Luca can often be found in Rome, New York, or Beijing.