Approximating Unique Games*

Anupam Gupta Kunal Talwar

Abstract

The UNIQUE GAMES problem is the following: we are given a graph $G = (V, E)$, with each edge $e = (u, v)$ having a weight w_e and a permutation π_{uv} on $[k]$. The objective is to find a labeling of each vertex u with a label $f_u \in [k]$ to minimize the weight of unsatisfied edges—where an edge (u, v) is *satisfied* if $f_v = \pi_{uv}(f_u)$.

The *Unique Games Conjecture* of Khot [8] essentially says that for each $\varepsilon > 0$, there is a k such that it is NP-hard to distinguish instances of Unique games with $(1-\varepsilon)$ satisfiable edges from those with only ε satisfiable edges. Several hardness results have recently been proved based on this assumption, including optimal ones for Max-Cut, Vertex-Cover and other problems, making it an important challenge to prove or refute the conjecture.

In this paper, we give an $O(\log n)$-approximation algorithm for the problem of minimizing the number of unsatisfied edges in any Unique game. Previous results of Khot [8] and Trevisan [12] imply that if the optimal solution has $\mathsf{OPT} = \varepsilon m$ unsatisfied edges, semidefinite relaxations of the problem could give labelings with $\min\{k^2\varepsilon^{1/5}, (\varepsilon \log n)^{1/2}\}m$ unsatisfied edges. In this paper we show how to round a LP relaxation to get an $O(\log n)$-approximation to the problem; i.e., to find a labeling with only $O(\varepsilon m \log n) = O(\mathsf{OPT} \log n)$ unsatisfied edges.

1 Introduction

There has been much recent interest in the *Unique Games conjecture*, first proposed by Khot [8]. To explain this conjecture, and our results, let us define the UNIQUE GAMES problem. An instance of this problem is a graph $G = (V, E)$, with each edge e having a weight w_e. We are also given a set of k labels, which we identify with the set $[k] = \{1, 2, \ldots, k\}$. Each edge $e = (u, v)$ in the graph comes equipped with a permutation $\pi_{uv} : [k] \to [k]$. The output of the problem is a labeling $f : V \to [k]$ that assigns a label to each vertex of G; an edge (u, v) is said to be *satisfied* under f if $f(v) = \pi_{uv}(f(u))$, else it is said to *violated* or *unsatisfied*.

Note that it is possible to define two natural optimization problems in this situation: we can seek to minimize the weight of the unsatisfied edges, as we do in this paper, or we can try to maximize the weight of satisfied edges. (These two objectives are equivalent from the viewpoint of exact optimization, but their approximability thresholds are quite different.) But before we talk about these issues, let us state the UGC.

CONJECTURE 1.1. (**Unique Games Conjecture [8]**) *For every $\varepsilon > 0$, there is a $k = k(\varepsilon)$ such that it is NP-hard to distinguish whether a Unique Game (with k labels) has a labeling that satisfies $(1 - \varepsilon)$ of the edges, or whether all labelings satisfy only ε fraction of the edges.*

Based on this conjecture, several hardness results have been proved: Khot and Regev showed that assuming the UGC, it is NP-hard to get a $(2 - \varepsilon)$-approximation algorithm for Vertex Cover [10], and Khot et al. [9] showed the hardness of getting an $(\alpha + \varepsilon)$-approximation algorithm for Max-Cut, where $\alpha = 0.87856\ldots$ is the approximation guarantee of the SDP-based Max-Cut algorithm of Goemans and Williamson [6]. Furthermore, Chawla et al. [3], and independently, Khot and Vishnoi [11] showed that MultiCut, Sparsest Cut and other cut problems are hard to approximate within any constant, assuming the Unique Games Conjecture.

Given the usefulness of the UGC to proving tight hardness results, it is natural to attempt to prove (or disprove) the conjecture. Indeed, this has spurred on some very interesting work. Feige and Reichman [5] showed that the Unique Games problem was NP-hard: they show that for every $\varepsilon > 0$, there is a $c > 0$ such that it is hard to distinguish whether c-fraction of the edges are satisfiable, or only $\varepsilon\, c$-fraction are satisfiable. (Note that this result does not prove the UGC, since the value of c is much less than $1 - \varepsilon$.) The result, however, does show that the problem of *maximizing* the number of satisfied edges in a unique game is $\Omega(2^{\log^{1-\delta} n})$-hard for every δ.

Our Results. In this paper, we consider the *minimization* version of the *Unique Games* problem, which

*This work was partly done while the first author was visiting Microsoft Research, Redmond; the first author's research is partly supported by an NSF CAREER award CCF-0448095, and by an Alfred P. Sloan Fellowship.

SODA '06, January 22-26, Miami, FL
 ISBN 0-89871-605-5/06/01

we call the Min-UniqGame problem. Given a unique game, we define the *cost* of a labeling f to be the weight of edges unsatisfied by f:

$$\mathsf{cost}(f) = \sum_{(u,v)\in E: f(v)\neq \pi_{uv}(f(u))} w_{uv} \qquad (1.1)$$

Let OPT denote the cost of the optimal labeling, and the goal is to get a labeling with cost not much more than OPT. Our main theorem is the following:

Theorem 1.1. *There is an $O(\log n)$-approximation algorithm for the* Min-UniqGame *problem.*

Our algorithm is based on rounding a linear-programming relaxation for the problem. We also show that the relaxation has an integrality gap of $\Omega(\log n)$, and hence our rounding is existentially tight.

The question of minimizing the number of violated edges in a unique games has been considered before: papers of Khot [8] and Trevisan [12] both give algorithms for unique games. (Both the algorithms are given for the unweighted case where $w_e = 1$.) In his original paper [8], Khot gave an SDP-rounding algorithm with the following guarantee: if the optimal solution has $\varepsilon\, m$ of the edges unsatisfied, then Khot's algorithm gives a labeling that has cost $O(\varepsilon^{1/5}\, k^2\, m)$.

In a recent paper [12], Trevisan gave a SDP-rounding algorithm with a slightly different guarantee, this time in terms of n. If the optimal labeling violates $\varepsilon\, m$ edges, then his algorithm violates $O((\varepsilon \log n)^{1/3})$ edges. Note that when $\varepsilon = \Theta(1/\log n)$, Trevisan's algorithm also gives an $O(\log n)$ approximation, but the approximation guarantee gets worse as ε becomes smaller. (It is not difficult to improve the bound to $O((\varepsilon \log n)^{1/2})$—see Section 4 for a short discussion—but we do not see how to obtain an approximation algorithm using his techniques.)

Our Techniques. In this paper, we solve an LP relaxation for the Min-UniqGame problem: while a simplistic LP has an arbitrary large integrality gap, we add some "cycle constraints" that ensure that if a vertex v gets a color c that, when propagated around some cycle back to v, results in a color $c' \neq c$, then some edge on the cycle must be violated. This LP gives us lengths $d(e)$ for the edges in G.

We first approximate the graph distances by distances in a tree T, such that the average distortion between the distances in G, and those in T is at most $O(\log n)$, and then give a natural "propagation" rounding procedure that ensures that the probability of any edge (u, v) being violated is at most $O(d(u, v) + d_T(u, v))$; on average, this is at most $O(\log n)d(u, v)$, and hence we get the claimed $O(\log n)$ approximation for Min-UniqGame.

The reader will note that while the previous algorithms were based on rounding a SDP relaxation, our algorithm merely uses an LP relaxation. However, it is not clear whether our linear program is indeed weaker, since it uses these additional "cycle constraints". Given that the integrality gap of our LP is $\Omega(\log n)$, it is an interesting problem to investigate whether adding these cycle constraints to the SDP would lead to better approximation algorithms.

2 Linear Programming Relaxation

Let us write an integer programming formulation for the problem as follows. For each node u, we introduce a set of k 0-1 variables $\{x(u, 1), x(u, 2), \ldots, x(u, k)\}$, where the variable $x(u, l)$ is set to one if and only if the node u is labeled with the label $l \in [k]$. Since each node has a unique label, we can enforce the constraint that $\sum_l x(u, l) = 1$.

Note that if an edge (u, v) is violated, the quantity $\sum_l |x(u, l) - x(v, \pi_{uv}(l))|$ is equal to two, and is zero otherwise. We introduce variables $d(u, v, l)$, along with the constraint $d(u, v, l) \geq |x(u, l) - x(v, \pi_{uv}(l))|$. Thus the objective function of the integer program is to minimize the total weight of violated edges

$$\sum_{(u,v)\in E} \frac{w_{uv}}{2} \sum_l d(u, v, l) \qquad (2.2)$$

Let us add valid "cycle" constraints to strengthen this formulation; these will be useful when we take the linear programming relaxation of the integer program. Let C be a simple cycle $u = v_0, v_1, \ldots, v_t = u$ in G containing u. Let l_0 be a label for v_0: for each value of $i \in [1, t]$, inductively define l_i as $l_i = \pi_{v_{i-1}v_i}(l_{i-1})$. In other words, the l_i's are defined so that $l_0, l_1, \ldots, l_i$ are labels that satisfy each of the edges $(v_0, v_1), \ldots, (v_{i-1}, v_i)$. Note that this process also defines another label l_t for $u = v_t$ which may or may not agree with the initial label l_0: indeed, we say that the label l_0 is *bad* for u with respect to C if $l_t \neq l_0$. Let $B_{u,C}$ be the set of labels that are bad for u with respect to cycle C. Note that for any labeling f, if the label $f(u) = l_0$ lies in $B_{u,C}$, there must be at least one position i such that the label $f(v_i) = l_i$ and the next label $f(v_{i+1}) \neq l_{i+1}$; i.e., there must be at least one edge (v_i, v_{i+1}) that is violated. Hence for every such cycle C and every label $l_0 \in B_{u,C}$, we can write a constraint $\sum_{i=1}^{t} d(v_{i-i}, v_i, l_{i-1}) \geq x(u, l_0)$.

This gives us the linear program in Figure 2.1; this is the LP that we will consider in the rest of the paper. While this linear program has an exponential number of constraints, it can be solved in polynomial time using the ellipsoid method given a polynomial-time separation oracle [7]. We now show such a separation oracle.

$$\text{minimize } Z^* = \sum_{(u,v)\in E} \frac{w_{uv}}{2} \sum_l d(u,v,l) \tag{LP}$$

$$\text{s.t.} \quad \sum\nolimits_l x(u,l) = 1 \qquad \forall u \in V \tag{2.3}$$

$$d(u,v,l) \geq |x(u,l) - x(v,\pi_{uv}(l))| \qquad \forall u,v \in V, l \in [k] \tag{2.4}$$

$$\sum\nolimits_{i=1}^{t} d(v_{i-1}, v_i, l_{i-1}) \geq x(u,l_0) \qquad \forall C, \forall u \in C, \forall l_0 \in B_{u,C} \tag{2.5}$$

$$0 \leq x(u,l) \leq 1 \qquad \forall u \in V, \forall l \in [k] \tag{2.6}$$

Figure 2.1: The LP relaxation

THEOREM 2.1. *Given a proposed solution* $(\mathbf{x}, \mathbf{d})$ *to the linear program (LP), there is a polynomial time procedure to output a violated constraint, if any.*

Proof. The constraints (2.3), (2.4) and (2.6) can be directly verified, since they are only polynomially many. It remains to check if one of the cycle constraints (2.5) is violated. Towards this goal, we create an auxiliary graph G' as follows. The nodes of G' are node-label pairs (u,l) for each $u \in V, l \in [k]$. We add an edge from (u,l) to (v,l') if $(u,v) \in E$ and $l' = \pi_{uv}(l)$ and assign it an edge-length of $d((u,l),(v,l')) = d(u,v,l)$. Let P be a simple path in G' starting at a node (u,l). We say that the path P is a *conflict* path if it is a path from (u,l) to (u,l') for $l \neq l'$. We say that such a path P is *elementary* if it does not contain $(v,\hat{l})$ and $(v,\hat{l}')$ as internal nodes, for any node $v \neq u$ and labels $\hat{l} \neq \hat{l}'$. Finally, we say that P is *short* if the sum of edge lengths on P is strictly smaller than $x(u,l)$.

We claim that there is a short elementary conflict path in G' if and only if there is violated cycle constraint in the LP. Indeed, any elementary conflict path P from (u,l) to (u,l') in G' naturally corresponds to a cycle C containing u such that $l \in B_{u,C}$; moreover, this path being short corresponds precisely to the cycle constraint (2.5) for $l \in B_{u,C}$ being violated. Conversely, if there is a violated constraint of the form (2.5), then consider the path P in G' that corresponds to starting from (u,l_0), following the vertices of the cycle until we see (u,l_t). This path would be a conflict path; furthermore, the constraint being violated implies that it would be short as well. Finally, the conflict path corresponding to a simple cycle is always elementary.

Thus it suffices to check for each (u,l), whether G' has a short elementary conflict path starting at (u,l). It is easy to find a short conflict path in G' using a shortest path algorithm: however, this short (u,l)-(u,l') path may not be *elementary*, and it seems difficult to find such an elementary cycle. However, the lemma below shows that if we find a short conflict path, we can find *some* short elementary conflict path, and hence find a cycle constraint that is violated.

LEMMA 2.1. *If G' has a short conflict path, then it contains a short elementary conflict path.*

Proof of Lemma 2.1. Let p be a short path from (u,l_u) to (u,l'_u). If p is elementary, we are done. If not, it contains an elementary conflict subpath p' from (v,l_v) to (v,l'_v) for some v, some $l_v \neq l'_v$. We now argue that p' must be short as well, i.e. $d(p') \leq x(v,l_v)$, where $d(p)$ denotes the length of a path in G' under the distance function d.

Since p was short, $d(p) \leq x(u,l_u)$. Let p'' be the subpath of p from (u,l_u) to (v,l_v). Clearly $d(p') \leq d(p) - d(p'')$. From constraints (2.4) and the triangle inequality, it follows that $d(p'') \geq |x(u,l_u) - x(v,l_v)|$. Thus $x(v,l_v) \geq x(u,l_u) - d(p'')$ and so it follows that $d(p') \leq x(v,l_v)$. This completes the proof of the lemma. ■

This gives us the claimed separation oracle. ■

3 Rounding the Linear Program

In this section, we will show how to round a given solution to the LP, and to construct a labeling $f : V \to [k]$ such that $E[\mathsf{cost}(f)] \leq O(\log n) \times Z^*$. The main idea of the rounding will be to interpret the fractional cost of an edge as the "length" of an edge. Let $x(u,l)$ be a solution to the above linear program. We now use it to get a length function $d : E \to \mathbb{R}$ on the edges: for an edge $(u,v) \in E$, let us define $d(u,v) = \sum_l d(u,v,l)$. Using this notion of distance, we can rewrite the objective function value of the LP solution as $Z^* = \frac{1}{2}\sum_{(u,v)\in E} w_{uv} d(u,v)$.

REMARK 3.1. *Note that the distance function d may not be a metric; i.e., the triangle inequality may be violated. E.g., consider an instance where G is a triangle K_3, and the label set is $\{0,1\}$. When each edge of the triangle has a not-equal constraint (i.e., $\pi_e =$* (12)

for each edge e) and all edges have weight 1, the optimal integer solution $L_u = L_w = 0, L_v = 1$ assigns length zero to the satisfied edges $(u,v),(v,w)$ and length one to the unsatisfied edge (u,w), thus violating the triangle inequality.

Let $\widehat{d}$ be the metric completion of d; i.e., the shortest-path distances between vertices in V according to the edge-lengths d. We can now use results of Fakcharoenphol et al. [4] to find a tree $T = (V, E_T)$ which approximates $\widehat{d}$ in the following sense:

$$d_T(u,v) \geq \widehat{d}(u,v) \qquad \forall\, u,v \in V \qquad (3.7)$$

$$d_T(u,v) = d(u,v) \qquad \forall\, (u,v) \in E_T \qquad (3.8)$$

$$\sum_{(u,v)\in E} w_{uv}\, d_T(u,v) \leq O(\log n) \sum_{(u,v)\in E} w_{uv}\, d(u,v) \qquad (3.9)$$

We will now show how to randomly round the LP solution guided by this tree such that the expected weight of the violated edges (i.e., the expected cost) is no more than of $\sum_{(u,v)\in E} w_{uv} 3(d_T(u,v) + d(u,v))$. Now using (3.9), this expected cost is at most $O(\log n) \times \sum_{(u,v)\in E} w_{uv}\, d(u,v) = O(\log n) \times Z^*$, and hence we will get an $O(\log n)$-approximation.

3.1 Propagation Rounding: The Subtree Case

The algorithm uses *propagation rounding*: we set the label for some nodes, and then "propagate" the results of this labeling out to the rest of the nodes. For simplicity of exposition, and to illustrate the ideas and intuition, let us first give an algorithm for the case when the tree $T = (V, E_T)$ is a subgraph of $G = (V, E)$; i.e., the edge set $E_T \subseteq E$.

The Algorithm. The propagation rounding scheme works as follows: we first pick an arbitrary vertex $u \in V$, and assign it a random label L_u according to the probability distribution defined by $x(u,l)$; i.e., $\mathbf{Pr}[L_u = l\,] = x(u,l)$. If v is a neighbor of u in T, we assign it a label L_v correlated to our choice for L_u so that the following properties hold:

1. The marginal probability $\mathbf{Pr}[L_v = l\,] = x(v,l)$.
2. The probability of (u,v) being violated is $\mathbf{Pr}[L_v \neq \pi_{uv}(L_u)\,] \leq d(u,v)$.

Such a correlated rounding can be done as follows: let $y(u,l,v,l')$ be a minimum cost transportation from the distribution $x(u,\cdot)$ to $x(v,\cdot)$, where the cost $c_{ulvl'}$ of transporting one unit of flow from (u,l) to (v,l') is zero if $l' = \pi_{uv}(l)$, and is one otherwise. Note that the cost of this transportation is

$$\begin{aligned}\sum_{l,l'} c_{ulvl'}\, y(u,l,v,l') &= \sum_{l,l' \neq \pi_{uv}(l)} y(u,l,v,l') \\ &\leq \sum_l |x(u,l) - x(v,\pi_{uv}(l))| \\ &\leq \sum_l d(u,v,l) = d(u,v). \qquad (3.10)\end{aligned}$$

If u gets the label L_u, we now pick a label L_v for v according to the probability distribution $\mathbf{Pr}[L_v = l'\,] = \frac{y(u,L_u,v,l')}{x(u,L_u)}$. And once v is labeled, all its unlabeled neighbors can be labeled similarly, allowing us to extend the rounding to the whole tree. We note that this rounding process is *symmetric*, i.e. the resulting distributions on the labelings of the nodes is independent of the starting node u.

The following lemma will allow us to bound the cost of the resulting random labeling.

LEMMA 3.1. *The probability that the graph edge $(u,v) \in E$ is violated is*

$$\mathbf{Pr}[L_v \neq \pi_{uv}(L_u)\,] \leq d(u,v) + 2\, d_T(u,v) \qquad (3.11)$$

Proof. For a tree edge $(u,v) \in E_T$, the probability that the edge is violated is precisely

$$\begin{aligned}&\textstyle\sum_{l' \neq \pi_{uv}(l)} \mathbf{Pr}[L_u = l\,]\mathbf{Pr}[L_v = l' \mid L_u = l\,] \\ &= \sum_{l' \neq \pi_{uv}(l)} x(u,l) \frac{y(u,l,v,l')}{x(u,l)} \leq d(u,v) \qquad (3.12)\end{aligned}$$

using the calculations in (3.10). Now let (u,v) be a non-tree edge, and let C be the cycle formed on adding (u,v) in T. The probability of the edge (u,v) being violated can be written as

$$\begin{aligned}&\mathbf{Pr}[(u,v) \text{ violated}\,] \\ &= \textstyle\sum_l \mathbf{Pr}[L_u = l\,]\, \mathbf{Pr}[(u,v) \text{ violated} \mid L_u = l\,] \\ &= \textstyle\sum_l x(u,l)\, \mathbf{Pr}[(u,v) \text{ violated} \mid L_u = l\,] \\ &= \textstyle\sum_{l \notin B_{u,C}} x(u,l)\, \mathbf{Pr}[(u,v) \text{ violated} \mid L_u = l\,] \\ &\qquad + \textstyle\sum_{l \in B_{u,C}} x(u,l)\, \mathbf{Pr}[(u,v) \text{ violated} \mid L_u = l\,]\end{aligned}$$

We bound each of the two sums separately. Let the tree path from u to v be $\langle u = v_0, v_1, \ldots, v_t = v\rangle$. When the label l is not in the bad set $B_{u,C}$, the edge (u,v) is satisfied if each of the edges (v_{i-1}, v_i) for $i = 1, \ldots, t$ is satisfied. And hence $\mathbf{Pr}[(u,v)$ is violated $\mid L_u \notin B_{u,C}\,]$ is at most the probability that one of the edges (v_{i-1}, v_i) being violated, which by (3.12) and a trivial union bound is at most $\sum_{i=1}^{t} d(v_{i-1}, v_i) \leq d_T(u,v)$.

We can bound the second term by $\sum_{l \in B_{u,C}} x(u,l)$, and want to show that this is at most the length of the cycle C. Indeed, let the cycle C have vertices $u = v_0, v_1, \ldots, v_t = v, v_{t+1} = u$. Given a label l_0^s for

u, define l_i^s so that each of the edges (v_{j-1}, v_j) for $j \le i$ is satisfied; i.e. $l_j^s = \pi_{v_{j-1}v_j}(l_{j-1}^s)$. Note that for each i, the map from l_0^s to l_i^s is a permutation.

Now for any label $l_0^s \in B_{u,C}$, the corresponding cycle constraint (2.5) implies that $x(u, l_0^s) \le \sum_{i=1}^{t+1} d(v_{i-1}, v_i, l_{i-1}^s)$. Hence summing over all labels in $B_{u,C}$, we get

$$\begin{aligned}
&\textstyle\sum_{l \in B_{u,C}} x(u,l) && (3.13)\\
&\le \textstyle\sum_{l_0^s \in B_{u,C}} \sum_{i=1}^{t+1} d(v_{i-1}, v_i, l_{i-1}^s)\\
&\le \textstyle\sum_{l_0^s \in [k]} \sum_{i=1}^{t+1} d(v_{i-1}, v_i, l_{i-1}^s)\\
&\qquad\qquad \text{(Summing over all } l_0^s\text{)}\\
&= \textstyle\sum_{i=1}^{t+1} \sum_{l_0^s \in [k]} d(v_{i-1}, v_i, l_{i-1}^s)\\
&\qquad\qquad \text{(Changing order of summation)}\\
&= \textstyle\sum_{i=1}^{t+1} \sum_{l_{i-1}^s \in [k]} d(v_{i-1}, v_i, l_{i-1}^s)\\
&\qquad\qquad \text{(Map } l_0^s \to l_{i-1}^s \text{ is permutation)}\\
&= \sum_{i=1}^{t+1} d(v_{i-1}, v_i) = d_T(u,v) + d(u,v) && (3.14)
\end{aligned}$$

Summing the two expressions gives us the desired bound (3.11). ∎

3.2 Propagation Rounding: The Non-Subtree Case We now extend the above argument to the case when the tree $T = (V, E_T)$ is not necessarily a subtree of G, and there may be edges $(u, v) \in E_T$ which do not belong to E: we call these *fake edges*. For each fake edge (u, v) in T, let P_{uv} be a shortest u-v path $\langle u = v_0, v_1, \ldots, v_s = v\rangle$ in the graph G equipped with the distance function d. (For each real edge $(u, v) \in E$, we define $P_{uv} = (u, v)$ itself.) We define a permutation π_{uv} for this *fake edge* in the natural way by composing the permutations along this path P_{uv}. Hence, this defines a permutation for each edge of T, and we now use the propagation rounding algorithm of the previous section using this set of permutations on T.

We now have to prove a result similar to Lemma 3.1, but now it is for the case of non-subtrees.

LEMMA 3.2. *The probability that the graph edge $(u, v) \in E$ is violated using the above algorithm is*

$$\mathbf{Pr}[L_v \ne \pi_{uv}(L_u)\,] \le 3\, d(u,v) + 2\, d_T(u,v) \qquad (3.15)$$

Proof. Consider any tree edge $(u, v) \in E_T \cap E$: the same analysis as in Lemma 3.1 shows that the probability that the edge is violated is at most $d(u, v)$. However, for a non tree edge $(u, v) \in E \setminus E_T$, the problem is that the cycle C formed by adding (u, v) to T may not exist in G, since some of the edges in the cycle C may be fake. In turn, this implies that we do not have a constraint in the LP corresponding to C, and we cannot use the same analysis as before. Before we start on the new analysis, it is useful to recall that the permutations on the fake edges are defined based on the LP solution and the tree.

Let $E_C \subseteq E$ be the multiset of graph edges obtained by taking the union of all paths P_{pq} corresponding to edges (p, q) in the cycle C. Since the length of the edges in the path P_{pq} is precisely $d_T(p, q) = d(p, q)$, we get that $\sum_{e \in E_C} d(e) = d_T(u, v) + d(u, v)$. Consider the set E_C: either it contains a u-v path P' which forms a cycle C' with (u, v) in the original graph, or the edge (u, v) is a cut-edge of E_C. For the latter case, we can imagine throwing in the "reverse" edge (v, u) into E_C, with $\pi_{vu} = \pi_{uv}^{-1}$: note that no labels are bad for this 2-cycle, and hence the LP does not change, but the rest of the argument will now be the same for the two cases.

Let this cycle C' be $u = v_0, v_1, \ldots, v_t = v, v_{t+1} = u$; remember that there is a cycle constraint (2.5) in the LP for each label $l \in B_{u,C'}$. Intuitively, these labels in $B_{u,C'}$ can be charged to $d(C')$ as before. The labels that induce a good labeling on the fake cycle C are paid for by $d_T(u, v)$. However, this may leave a third set of labels which need to be paid for by other cycles, not necessarily involving u. We formalize this below.

Case I: Labels in $B_{u,C'}$. As before, for any label l_0^s assigned to u, let l_i^s be the induced labels on C'; i.e., $l_i^s = \pi_{v_{i-1},v_i}(l_{i-1}^s)$. For $l_0^s \in B_{u,C'}$, the analysis of (3.14) still works and one can infer that $\sum_{l_0^s \in B_{u,C'}} x(u, l_0^s)$ is at most $d(P') + d(u, v)$. Since the edges in P' are a subset of edges in E_C, this is bounded by $d_T(u, v) + d(u, v)$.

Case II: Labels not in $B_{u,C'}$. Let us now consider labels $l_0^s \notin B_{u,C'}$. Recall that E_C consisted of all the edges on the paths P_{pq} corresponding to edges $(p, q) \in C$. Let V_C be the set of vertices spanned by these edges E_C, and let T_C be a spanning tree of this subgraph (V_C, E_C) such that T_C contains all of the path P'. Recall that since we started off with labeling u with $l_0^s \notin B_{u,C'}$, we have a consistent labeling for the cycle C' (and hence for the path P' as well). We try to extend this labeling l_i^s of P' to all the vertices in V_C in the following fashion. We perform a breadth first traversal of the tree T_C: if we reach an unlabeled node q in $V_C \setminus P'$ from its labeled parent p (which has label l_p^s), we assign $l_q^s = \pi_{pq}(l_p^s)$. If this labeling violates a back-edge $e_b = (q, r) \in E_C \setminus T_C$, we stop and add l_0^s to a new set A. If there are no violations, we terminate with a consistent labeling for all of V_C; in this case we add l_0^s to a set A'. Note that $A \cup A' = [k] \setminus B_{u,C'}$; we will consider the two cases separately. In both cases, let l_p^s be the (possibly) partial labeling obtained when we start with l_0^s.

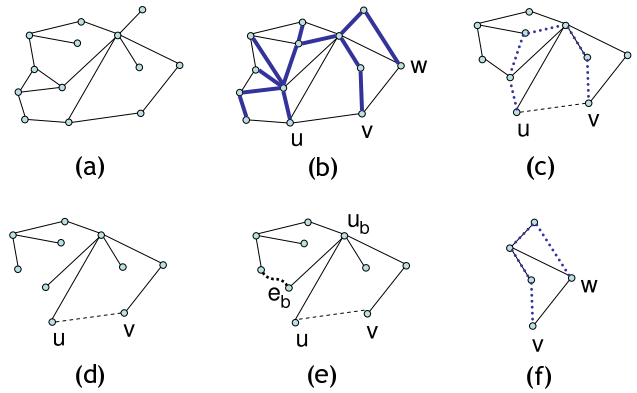

Figure 3.2: (a) Graph G. (b) Tree T on V. (c) The subgraph (V_C, E_C) for edge (u, v). (d) The tree T_C and cycle C.(e) Edge e_b and cycle C_b. (f) Subgraph (V_C, E_C) for edge (v, w). In this case, (v, w) is a cut edge for (V_C, E_C).

1. Suppose $l_0^s \in A$, and we stop when some back-edge $e_b = (q, r) \in E_C \setminus T_C$ is violated. Let C_b be the cycle obtained by adding the edge e_b to the spanning tree T_C, and let u_b be the first node on C_b to get labeled. Clearly, the label $l^s_{u_b} \in B_{u_b, C_b}$, else there would not be a violation on this cycle. Hence, by the cycle constraints in the LP,

$$x(u_b, l^s_{u_b}) \leq \textstyle\sum_{(p,q)\in C_b} d(p, q, l_p^s). \tag{3.16}$$

Moreover, if P_b is the path in the tree T_C from u to this first node u_b, then by constraints (2.4) and the triangle inequality,

$$x(u, l_0^s) \leq x(u_b, l^s_{u_b}) + \textstyle\sum_{(p,q)\in P_b} d(p, q, l_p^s). \tag{3.17}$$

Since the path P_b is disjoint from the cycle C_b, and their union is contained within E_C, we can sum up (3.16) and (3.17) to get

$$x(u, l_0^s) \leq \textstyle\sum_{(p,q)\in E_C} d(p, q, l_p^s). \tag{3.18}$$

Now we can sum up (3.18) for the labels $l_0^s \in A$ to get

$$\begin{aligned}&\textstyle\sum_{l_0^s \in A} x(u, l_0^s)\\ &\leq \textstyle\sum_{l_p^s \in [k]} \sum_{(p,q)\in E_C} d(p, q, l_p^s)\\ &= \textstyle\sum_{(p,q)\in E_C} d(p, q) \leq d(u, v) + d_T(u, v).\end{aligned}$$

2. Suppose $l_0^s \in A'$, and we terminate with a consistent extension of the labeling l_0^s to all the vertices in V_C so that none of the edges in E_C are violated. Since l_0^s was not a bad label for u with respect to C', and this labeling induced by l_0^s can be extended to all of E_C, the label l_0^s is not bad with respect to the fake cycle C either. Thus for (u, v) to be violated, at least one of the fake edges on the tree path must be violated. But by the properties of the propagation rounding, each fake edge (p, q) is violated with probability at most $d(p, q)$; using the union bound, the chance that (u, v) is violated is now at most $\sum_{(p,q)\in C\setminus\{(u,v)\}} d(p, q) = d_T(u, v)$.

Hence the total probability that the edge (u, v) is violated is at most $3d(u, v) + 2d_T(u, v)$, hence proving the lemma. ∎

3.3 The Integrality Gap of the LP Relaxation

In this section we show that the integrality gap of the LP relaxation is $\Omega(\log n)$. This shows that the algorithm we give is existentially tight, and we would have to use other techniques to obtain better results.

Let $G = (V, E) \sim G(n, \frac{d}{n})$ be a random graph where each edge is present with probability d/n. The following result is a well-known application of the probabilistic method; see, e.g., [2, pp. 38–39].

LEMMA 3.3. *There is a universal constant $c_d = c_d(d)$, such that with probability $(1 - o(1))$, G has at most n cycles of length less than $c_d \log n$.*

Moreover, we show that with high probability, G is *far from* being bipartite.

LEMMA 3.4. *With probability $(1 - o(1))$, $2n$ edges need to be deleted from G to make it bipartite, for $d \geq 50$.*

Proof. Let V_1, V_2 be a potential bipartition of V, and let us call such a bipartition (V_1, V_2) *easy* if there are at most $2n$ *non-cut* edges in G (i.e., V_i-V_i edges for some $i \in \{1, 2\}$). It suffices to show that with high probability, there are no easy bipartitions.

For a given bipartition, the number of potential non-cut edges is at least $\binom{n}{2} - n^2/4 = n(n-2)/4 \geq n^2/5$. Since each edge is present with probability $\geq 50/n$, the expected number of non-cut edges is at least $10n$. Thus by a Chernoff bound, the probability that there are at most $2n = (1 - \frac{4}{5})10n$ non-cut edges in G (and hence that the bipartition is *easy*) is at most $\exp\{-\frac{(4/5)^2 \cdot (10n)}{(2+(4/5))}\} \leq \exp(-n)$. Finally, since the number of potential bipartitions is at most 2^n, a trivial union bound implies the probability of an easy bipartition existing is at most $(2/e)^n = \exp(-cn)$, which proves the lemma. ∎

Consider a graph G that satisfies the conditions specified in both Lemma 3.3 and Lemma 3.4. Since G has at most n *short* cycles (i.e., cycles of length less

than $g \stackrel{def}{=} c_d \log n$), we now delete one edge from each short cycle to obtain a graph G' with *girth* g. In the process, we have deleted at most n edges, and hence G' still must be n edges far from being bipartite. Moreover, with probability $1 - o(1)$, G' has at most $50n$ edges.

We construct our MIN-UNIQGAME instance on G' as follows. The label set is just the set $\{1, 2\}$. Each edge (u, v) of G' has a constraint $L_u \neq L_v$ with weight 1. (Hence this is an instance of the MIN-UNCUT problem.) Let us first show that the LP solution has a small value.

LEMMA 3.5. *There exists a fractional solution to the linear program (LP) on the above instance G' with $Z^* = O(n/g)$.*

Proof. For each node v, let us pick the value of $x(u, 1)$ to be one of the two values $\{\frac{1}{2} + \frac{3c}{g}, \frac{1}{2} - \frac{3c}{g}\}$ uniformly at random; here c is a constant to be fixed later. Set $d(u, v, l)$ to be exactly $|x(u, l) - x(v, \pi_{uv}(l))|$. We show that this solution is feasible with high probability.

Consider a pair of nodes (u, v) at graph distance $g/3$ and let p_{uv} be the shortest path between them. The random choice of values ensures that each edge has expected length $\frac{3c}{g}$, and hence the expected length of the path p_{uv} is c. Thus, with probability $1 - \exp(-(c - 1)^2 g/(2c - 1))$, the LP solution assigns a distance of at least 1 to the path p_{uv}. Taking c to be large enough, this probability is at least $(1 - \frac{1}{n^3})$. Since there are at most n^2 such pairs, with probability $(1 - \frac{1}{n})$, every such path p_{uv} has at least unit length. The graph G has girth g, and hence each cycle in G' contains at least three such paths, making its LP length at least three. It follows that all cycle constraints are satisfied and hence this solution is feasible. Finally, since the number of edges is linear and each edge has length $O(\frac{1}{g})$, the cost of the LP is $O(n/g)$, as claimed. ∎

Next we show that optimal solution must violate a constant fraction of the constraints.

LEMMA 3.6. *The optimal solution for the above instance has cost at least n.*

Proof. For each $i \in \{1, 2\}$, let V_i be the set of nodes labeled i in the optimal solution. Then since every V_i-V_i edge is violated, the cost of the optimal solution is at least n. ∎

The integrality gap of $\Omega(g)$ follows from Lemma 3.5 and Lemma 3.6 above.

4 A Brief Sketch of Trevisan's Construction

In a recent paper [12], Trevisan essentially studied the problem of finding good algorithms for MIN-UNIQGAME. In a comment to the above paper, he gives an SDP-based algorithm for MIN-UNIQGAME where all edges have unit weights. His result can be paraphrased as follows:

THEOREM 4.1. *There is an algorithm which, given an instance of* MIN-UNIQGAME *on which the optimum labeling violates at most $O(\varepsilon^3/\log n)\, m$ edges, outputs a labeling with at most $\varepsilon\, m$ violated edges.*

In the following, we sketch (a minor modification of) his analysis, in which we show that if OPT violates $\delta\, m = O(\varepsilon^2/\log n)\, m$ edges, we can find a labeling with at most $\varepsilon\, m$ violated edges.

SDP Relaxation. Trevisan's algorithm first solves an SDP relaxation of the problem that has vectors $\mathbf{u}_l$ for each vertex $u \in V$ and label l such that $\sum_l \|\mathbf{u}_l\|^2 = 1$. The SDP also enforces the triangle inequality between all the vectors $\mathbf{u}_l$ (and the zero vector). Again, we think of this as defining a length $d(u, v) = \sum_l \|\mathbf{u}_l - \mathbf{v}_{\pi_{uv}(l)}\|^2$ for each edge (u, v) in V. The objective function of the SDP is $Y^* = \sum_{u \sim v} d(u, v)$, which by our assumption on the optimal value is at most $\delta\, m$.

The Rounding and Analysis. To get the claimed bound, we can delete all "long" edges with length $d(u, v) > \varepsilon/4$. Note that this causes us to violate at most $(4/\varepsilon)Y^* \leq (4/\varepsilon)\delta m = O(\varepsilon\, m/\log n)$ edges. We use a by-now-standard region growing technique (see, e.g., [13]) which guarantees the following:

THEOREM 4.2. *Given a graph $G = (V, E)$ with edge lengths d_e (such that $\sum_e d_e \geq Y^*$), and a parameter Δ, it is possible to decompose the graph into clusters $C_1, C_2, \ldots, C_t$ such that*

- *Each cluster C_i has a center vertex r_i, such that the shortest-path distance of every vertex in C_i (according to the edge lengths) is at most Δ, and*
- *the total number of edges cut is at most $\frac{O(\log n) \times Y^*}{\Delta}$.*

Setting $\Delta = \varepsilon/4$, we get that the number of edges cut is at most $Y^* \times O(\log n)/(\varepsilon/4) \leq \varepsilon\, m$.

Finally, for each cluster C with center r, Trevisan's algorithm assigns the label l to the center r with probability $\|\mathbf{r}_l\|^2$, and for each other vertex v in C, assigns it the label l' such that $\mathbf{v}_{l'}$ is the closest of v's vectors to that $\mathbf{r}_l$. Note that, given any edge (u, v) in cluster C, there is a path of length $\varepsilon/2$ from r to both the vertices (namely the shortest-path P_u from r to u, and this path P_u concatenated with the edge (u, v) to v). The simple and elegant analysis in that paper shows that the probability that either of these two paths are inconsistent is at most their lengths (and thus at most $\varepsilon/2$). Finally applying a union bound gives us that the

edge (u, v) is consistent with probability $1 - \varepsilon$, thus giving at most $\varepsilon\, m$ violated edges in this final rounding too. This completes the proof that at most $O(\varepsilon\, m)$ edges are violated during the run of the algorithm.

4.1 Conclusions and Future Directions In this paper, we gave an $O(\log n)$ approximation to the Min-UniqGame problem; we also showed that the LP relaxation has an $\Omega(\log n)$ integrality gap. It remains to be seen if the SDP relaxation (possibly with the cycle constraints that we added) can indeed give us a better guarantee. (Indeed, note that for the case $k = 2$, the Min-UniqGame problem is just the Min-UnCut problem, for which there is an $O(\sqrt{\log n})$ approximation due to Agarwal et al. [1], and it remains open to extend this to the case of general k.)

Acknowledgments We would like to think Shuchi Chawla, Kedar Dhamdhere, and Uri Feige for very helpful discussions, and to Moses Charikar for bringing Trevisan's paper [12] to our notice.

References

[1] A. Agarwal, M. Charikar, K. Makarychev, and Y. Makarychev, *$o(\sqrt{logn})$ approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems*, in Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, New York, NY, USA, 2005, ACM Press, pp. 573–581.

[2] N. Alon and J. Spencer, *The Probabilistic Method*, Wiley Interscience, New York, 1992.

[3] S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, and D. Sivakumar, *On the hardness of approximating multicut and sparsest-cut*, in Proceedings of the 20th IEEE Annual Conference on Computational Complexity, 2005.

[4] J. Fakcharoenphol, S. Rao, and K. Talwar, *A tight bound on approximating arbitrary metrics by tree metrics*, in Proceedings of the thirty-fifth ACM symposium on Theory of computing, ACM Press, 2003, pp. 448–455.

[5] U. Feige and D. Reichman, *On systems of linear equations with two variables per equation*, in Proceedings of the 7th APPROX, vol. 3122 of Lecture Notes in Computer Science, 2004, pp. 117–127.

[6] M. X. Goemans and D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. Assoc. Comput. Mach., 42 (1995), pp. 1115–1145.

[7] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*, Springer-Verlag, Berlin, 1988.

[8] S. Khot, *On the power of unique 2-prover 1-round games*, in 34th Annual ACM Symposium on the Theory of Computing, July 2002, pp. 767–775.

[9] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell, *Optimal inapproximability results for max-cut and other 2-variable csps?*, in Proceedings of the 45th Symposium on the Foundations of Computer Science (FOCS), 2004, pp. 146–154.

[10] S. Khot and O. Regev, *Vertex cover might be hard to approximate to within $2 - \varepsilon$*, in Proceedings of the 18th IEEE Annual Conference on Computational Complexity, 2003, pp. 379–.

[11] S. Khot and N. Vishnoi, *The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into ℓ_1*, in Proceedings of the 46th Symposium on the Foundations of Computer Science (FOCS), 2005, p. to appear.

[12] L. Trevisan, *Approximation algorithms for unique games*, Tech. Report TR05-034, ECCC, April 2005. See also the attached comment, May 13, 2005.

[13] V. V. Vazirani, *Approximation algorithms*, Springer-Verlag, Berlin, 2001.