Instituto de Computação – UNICAMP
Algoritmos de Aproximação
http://www.ic.unicamp.br/~lehilton/mo418a/

Unique Games Conjecture

Escrito por: Felipe L. De Mello, Victor F. Ferrari e Vinícius C. Espindola **Data:** 14/11/2019

1 Introdução

→ Vimos anteriormente um algoritmo de aproximação para o problema Max-Cut.

- Utilizava Semidefinite Programming
- .878-aproximação, chamada de algoritmo de *Goemans-Williamson*

→ Ao final da aula, foi visto um teorema: esse algoritmo é a melhor aproximação possível para o problema, assumindo a *Unique Games Conjecture*.

→ O que é a Unique Games Conjecture?

1.1 Histórico

Um aluno de pós-graduação chamado Subhash Khot ficou curioso para saber uma propriedade do problema de coloração em grafos. Já era sabido que decidir se um grafo é 3-colorível é NP-Difícil, mas a dúvida era se aumentar o número de cores disponíveis facilita o problema. Ou seja, se é mais fácil o problema de decidir se necessita de 3 cores, ou K, para qualquer constante K.

A partir desse problema, chegou à conclusão de que deveria estudar a complexidade do problema Unique Games, o que faz sentido ao olhar sua representação em grafo. Assim, em 2002, foi proposta a *Unique Games Conjecture*. História completa pode ser vista em [3].

2 Unique Games e Unique Label Cover

2.1 Unique Games

→ A **conjectura** do Unique Games é criada a partir do **problema** chamado Unique Games.

→ O Unique Games é um problema de satisfação de restrições, ou seja, é uma versão específica do CSP (Constraint Satisfaction Problem).

→ O CSP é dado por:

- Entrada:
  - Universo U de valores;
  - Variáveis $X_i \in U, \forall i \in \{1 \dots n\}$;
  - restrições $f : U^k \to \{0, 1\}$
- Solução: Atribuição de um valor em U para cada variável.
- Objetivo: maximizar o número de restrições satisfeitas.

→ Há a versão com pesos também.

→ Exemplos de problemas CSP: MAX-CUT, MAX-SAT.

- → O Unique Games é um CSP *binário*, ou seja, cada restrição corresponde a uma função em *duas variáveis*.

- → Além disso, para cada valor de U de uma das variáveis da restrição, há *exatamente um* valor para a outra variável que a satisfaz. Assim, uma restrição corresponde a uma função *bijetora*.

> **Conjectura 1** (Unique Games Conjecture: UGC)**.** *Dados quaisquer $\varepsilon, \delta > 0$, existe algum $k > 0$ dependente de ε e δ, tal que para o problema Unique Games com universo de tamanho k, é* NP-Difícil *distinguir entre instâncias nas quais pelo menos uma fração de $1-\varepsilon$ das restrições pode ser satisfeita, e instâncias nas quais no máximo uma fração de δ das restrições pode ser satisfeita.*

- → Informalmente, a UGC diz que é NP-Difícil diferenciar uma instância do problema na qual *quase todas* as restrições são satisfeitas e uma em que *quase nenhuma* é satisfeita.

- → Um problema é chamado *UG-Difícil* se ele é NP-Difícil considerando a UGC.

- → Não se sabe se há algoritmo de aproximação para o Unique Games que garanta desempenho que refute a UGC.

2.2 Unique Label Cover

- → Como as restrições do Unique Games são funções de duas variáveis, é fácil perceber que há uma representação do problema como *grafo*.

- → A versão em grafo do problema possui um nome: *Unique Label Cover*.

- → Essa versão é tão comum, e mais fácil de entender, que em muitos lugares o Unique Games é apresentado diretamente com ela!

- → Para a transformação do UG para o ULC, criamos *permutações*.
  - Uma permutação é um rearranjo do universo U do problema que mapeia, para cada restrição, o valor de uma variável ao valor da outra tal que $f(X_i, X_j) = 1$.
  - Isso pode ser feito pois a função é bijetora, e para universo de tamanho k, $U = 1 \dots k$.
  - Assim, podemos definir a permutação como $\pi(i) = j$ se $f(i, j) = 1$.

Assim, a transformação é:

Transformação-UG-ULC:

Crie um grafo vazio não-direcionado $G = (V, E)$
Para cada variável $X_u, u \in 1 \dots n$, insira o vértice u em V
Para cada restrição $f(X_u, X_v)$, insira a aresta (u, v) em E
Para toda aresta (u, v), crie uma permutação $\pi_{uv} : U \to U$ tal que $\pi_{uv}(i) = j$ se $f(i, j) = 1$
Retorne G, π

- → Assim, o problema se torna um de encontrar *labels (rótulos)* em vértices tais que a maior quantidade de permutações são satisfeitas.

- $\rightarrow$ Podemos verificar em tempo polinomial se *todas* as arestas do grafo são satisfatíveis.
  - Para toda componente conexa do grafo, teste todos os *labels* para um vértice arbitrário.
  - Para cada escolha, *propague* para todos os outros, pelas permutações.
  - Se todas as arestas são satisfatíveis, há algum *label* que gera uma atribuição perfeita, e isso pode ser verificado em tempo polinomial, no pior caso.
- $\rightarrow$ Similarmente, saber se *nenhuma* aresta é satisfatível também é trivial.
- $\rightarrow$ Porém, o mesmo não pode ser dito para todos os outros casos.
- $\rightarrow$ Pela UGC, se é desejado saber se uma **fração constante** de arestas é satisfatível, o problema é NP-Difícil para algum universo, independentemente da fração.
- $\rightarrow$ Existe um algoritmo de aproximação de fator $1-\sqrt{\varepsilon \log n}$ para o Unique Games/Unique Label Cover, baseado em *Semidefinite Programming*.
  - Esse algoritmo funciona para instâncias nas quais uma fração de $1-\varepsilon$ das arestas/restrições são satisfatíveis.
  - Se $\varepsilon \in O(1/\log n)$, essa aproximação é constante.
  - Não iremos mostrar esse algoritmo hoje, mas está especificado e demonstrado em [5].
- $\rightarrow$ Há também algoritmos *subexponenciais* para o problema e outros relacionados, ao contrário de diversos outros problemas NP-Difíceis. Alguns podem ser vistos em [1].

3 Consequências da UGC

- $\rightarrow$ Desde 2002, a UGC teve diversas aplicações em problemas relacionados, como pode ser visto na figura 1.
- $\rightarrow$ A figura 1 foi extraída de [4], que também possui diversos desenvolvimentos e conclusões em cima da UGC.
- $\rightarrow$ Como visto na figura, o principal uso da UGC foi para provas de *inaproximabilidade*, muitas vezes junto com outra técnica, como PCP.
- $\rightarrow$ Ela também deu origem a diversas *variantes*.
- $\rightarrow$ Deste ponto em diante, veremos duas reduções para problemas que provam inaproximabilidade ou que uma aproximação conhecida é a melhor possível.

4 Redução para Multicut

- $\rightarrow$ Relembrando o problema do multicorte:
  - Entrada:
    - grafo G=(V,E);
    - custo das arestas $c_e \geq 0$, $e \in E$;
    - pares de vértices fonte-ralo $s_1 - t_1, \ldots, s_k - t_k$, $s, t \in V$.

Problem	**Best Approx. Known**	**Inapprox. Known Under UGC**	**Best Inapprox. Known**	**Ref.**
Vertex Cover (VC)	2	$2-\varepsilon$	1.36	[68, 15, 36]
VC on k-uniform Hypergraphs, $k \geq 3$	k	$k-\varepsilon$	$k-1-\varepsilon$	[68, 16, 34]
MaxCut	α_{MC}	$\alpha_{\mathsf{MC}}-\varepsilon$	$\frac{17}{16}-\varepsilon$	[43, 63, 51] [87, 66]
Max-2SAT*	α_{LLZ}	$\alpha_{LLZ}-\varepsilon$	APX-hard*	[78, 12]
Any CSP $\mathcal{C}$ with integrality gap $\alpha_{\mathcal{C}}$	$\alpha_{\mathcal{C}}$	$\alpha_{\mathcal{C}}-\varepsilon$		[90]
Max-kCSP	$O(2^k/k)$	$\Omega(2^k/k)$	$2^{k-O(\sqrt{k})}$	[23, 101, 13, 100]
Max-3CSP on satisfiable instances	$\frac{8}{5}$	$\frac{8}{5}-\varepsilon$, under Conj. 3.6	$\frac{27}{20}-\varepsilon$	[105, 88, 69]
Max Acyclic Subgraph	2	$2-\varepsilon$	$\frac{66}{65}-\varepsilon$	[48, 86]
Feedback Arc Set	$\tilde{O}(\log N)$	$\omega(1)$	APX-hard	[48, 102]
Non-uni. Sparsest Cut	$\tilde{O}(\sqrt{\log N})$	$\omega(1)$	APX-hard	[8, 25, 71, 31]
Uniform Sparsest Cut,	$O(\sqrt{\log N})$	$\omega(1)$, under Hypo. 3.4	No PTAS	[10, 7, 3]
Min-2SAT-Deletion, Min-Uncut	$O(\sqrt{\log N})$	$\omega(1)$	APX-hard	[1, 60]
Coloring 3-colorable Graphs	$N^{.2111}$	$\omega(1)$, under Conj. 3.7	5	[6, 35, 58]
Coloring $2d$-colorable Graphs, $d \geq 2$	$N^{1-\frac{3}{2d+1}}$	$\omega(1)$, under Conj. 3.6	$2d+2\lfloor\frac{2d}{3}\rfloor-1$, $d^{\Omega(\log d)}$	[55, 35] [58, 59]
Scheduling with Prec. Constraints*	2	$2-\varepsilon$, under Hypo. 3.5		[15]
Kernel Clustering kernel matrix B	$K(B)$	$K(B)-\varepsilon$	APX-hard	[64, 65]
L_p Grothendieck Prob.* $p>2$	γ_p^2	$\gamma_p^2-\varepsilon$		[72] + Follow-up.
Multiway Cut integrality gap $\alpha \leq 1.344$	α	$\alpha-\varepsilon$	APX-hard	[21, 56, 81]

Figura 1: Reduções obtidas a partir da UGC.

- Solução: conjunto de arestas F.
- Objetivo: minimizar o número de arestas $|F|$ nas quais ao serem removidas desconectam todos os pares $s_1 - t_1, \ldots, s_k - t_k$.

Teorema 1. *Assumindo a conjectura do Unique Games, para qualquer constante $\alpha \geq 1$, não existe α-aproximação para o problema do multicorte a não ser que* $\mathrm{P} = \mathrm{NP}$.

→ Pelo teorema, é *UG-Difícil* aproximar o problema do multicorte por qualquer constante maior ou igual a 1.

→ Para a redução da UG para multicorte utilizamos um caso especial da UG chamada MAX 2LIN(k).

→ MAX 2LIN(k):

- Entrada:
  - L=0,...,k-1, $k \in \mathbb{Z}$;
  - Variáveis (vértices) $\in V$;
  - Restrições (arestas) $\in E$;
  - $\forall uv \in E$, temos $c_{uv} \in L$ tal que $\pi_{uv}(i) = i - c_{uv} (mod\, k)$, ou seja, uv é satisfeita $\iff$ os vértices u e v recebem rótulos i, j tais que $i - j = c_{uv} (mod\, k)$;
- Solução: Atribuição de rótulos R, tal que $\forall u \in V$, existe um rótulo $r_u \in L$;
- Objetivo: Maximizar número de arestas $uv \in E$ satisfeitas.

Conjectura 2 (Linear Unique Games Conjecture: LUGC)**.** *Dados quaisquer $\varepsilon, \delta > 0$, existe algum $k > 0$ dependente de ε e δ, a versão do unique games MAX 2LIN(k) com L=0,...,k-1, é* NP-Difícil *distinguir entre instâncias nas quais pelo menos uma fração de $1 - \varepsilon$ das arestas pode ser satisfeita, e instâncias nas quais no máximo uma fração de δ das arestas pode ser satisfeita.*

→ Para provar o *Teorema 1* são necessários 2 lemas:

Lema 1. *Para qualquer ε tal que $0 \leq \varepsilon \leq 1$, dado uma solução viável de uma instância de MAX 2LIN(k) que satisfaz pelo menos $(1-\varepsilon)|E|$ arestas, então existe uma solução viável para uma instância do multicut com custo de no máximo $\varepsilon|E'|$.*

Prova do Lema 1. Reduzir MAX 2LIN(k) para multicorte:

- Seja I uma instância do MAX 2LIN(k) com grafo $G = (V, E)$, universo L de rótulos de tamanho k e C o conjunto de pesos das arestas de E;
- Faça uma instância I' do multicorte da seguinte maneira:
  - G'=(V',E') com V' = V x L;
  - Arestas em E' entre pares vértice-rótulo (u,i) e (v,j) $\iff$ $uv \in E$ e $i - j = c_{uv} (mod\, k)$;
  - Também faça os pares fontes-ralo serem s=(u,i) e t=(u,j) para todo u $\in V$ e $i \neq j$;

– Note que $E' = k|E|$ e $V' = k|V|$;

– Suponha uma rotulação $x_u \in L$ de G que satisfaz pelo menos $(1-\varepsilon)|E|$ arestas de G;

– Particione V' em k partes, $V'_0, \ldots, V'_{k-1}$, onde $V'_c = \{(u, x_u + c(mod\, k))\} \forall u \in V$, ou seja, a c-ésima parte será o conjunto de vértices cuja rotulação satisfazem uma aresta de custo c;

– Note que s=(u,i) e t=(u,j) $\forall u \in V$ e $i \neq j$; estão em diferentes partes da partição, portanto, ao remover todas arestas com extremos em diferentes partes da partição obtemos uma solução do multicorte;

– Agora determinamos o custo desta solução. Considere qualquer aresta $((u,i),(v,j)) \in E'$ tal que (u,i) e (v,j) estão em diferentes partes da partição. Demonstraremos que a aresta (u,v) no grafo original não é satisfeita pelo rotulamento dado;

– Pela construção de E' sabemos que $i - j = c_{uv}(mod\, k)$. Também sabemos que (u,i) e (v,j) estão em diferentes partes da partição;

– Suponha que $(u,i) \in V_c$ e $(v,j) \in V_c$' para $c \neq c'$. Então $i = x_u + c(mod\, k)$ e $j = x_v + c'(mod\, k)$, e portanto:

$$\begin{aligned} c_{uv} &= i - j(mod\, k) \\ &= (x_u + c) - (x_v + c')(mod\, k) \\ &= (x_u - x_v) + (c + c')(mod\, k) \\ &\neq x_u - x_v(mod\, k) \end{aligned}$$

pois $c \neq c'$;

– Como uma aresta é satisfeita $\iff c_{uv} = i - j(mod\, k)$, isto significa que arestas entre vértices de diferentes partições não são satisfeitas. Como no máximo $\varepsilon|E|$ arestas não são satisfeitas em MAX 2LIN(k), no máximo $\varepsilon k|E| = \varepsilon|E'|$ arestas são removidas no multicorte.

□

Lema 2. *Para qualquer ε tal que $0 \leq \varepsilon \leq 1$, dado uma solução de uma instância do multicorte de custo máximo $\varepsilon|E'|$, então existe uma solução para uma instância do MAX 2LIN(k) que satisfaz pelo menos (1-2ε)$|E|$ arestas.*

Prova do Lema 2. omitida por simplificação, mas é o caminho inverso da prova do Lema 1. Pode ser vista em [5]. □

Prova do Teorema 1.

– Suponha que existe α-aproximação para o problema do multicorte;

– Então utilizando este algoritmo e o Lema 1, sabemos que: dado uma instância do MAX LIN2(k), na qual pelo menos (1-ε)$|E|$ arestas são satisfeitas, podemos encontrar uma solução do multicorte de custo $\varepsilon\alpha|E'|$;

– Pelo Lema 2 sabemos obter uma solução do MAX 2LIN(k), a partir deste multicorte, na qual pelo menos (1-2$\varepsilon\alpha$)$|E|$ arestas satisfeitas;

– Se a instância do MAX LIN2(k) satisfaz no máximo $\delta|E|$ arestas, então este algoritmo satisfaz no máximo $\delta|E|$ arestas;

- Fazendo $\varepsilon < \frac{1-\delta}{2\alpha}$, então temos que $(1-2\varepsilon\alpha)|E| > \delta|E|$.
- Isto implica que nosso algoritmo consegue distinguir entre instâncias nas quais pelo menos (1-ε)$|E|$ arestas são satisfeitas e instâncias nas quais no máximo $\delta|E|$ arestas são satisfeitas;
- Dado a UG, isto implica que P = NP.

□

5 Redução para Max-Cut

→ Relembrando o problema Max-Cut:

- Entrada:
  * grafo G=(V,E);
  * pesos $\omega : E \to \mathbb{R}_+$
- Solução: corte $[S,T], V = S \cup T$.
- Objetivo: max $\sum_{(i,j)\in E[S,T]} \omega_{ij}$
- $E[S,T]$ são arestas (i,j) com $i \in S$ e $j \in T$.

Teorema 2. *Assumindo a conjectura do Unique Games, não existe α-aproximação para o problema do corte máximo com constante*

$$\alpha > \min_{-1\leq x\leq 1} \frac{\frac{1}{\pi}\arccos x}{\frac{1}{2}(1-x)} \geq .878$$

a não ser que P = NP.

→ Ou seja, queremos provar que o algoritmo de Goemans-Williamson possui o *melhor fator de aproximação* para o Max-Cut, dada a UGC.

→ É, então, UG-Difícil aproximar o problema por um fator melhor que .878.

→ A prova completa é muito complexa, envolve conceitos um pouco mais específicos de *PCP*, então as ideias serão resumidas.

→ Esse mesmo método é utilizado para provar diversos outros resultados em relação ao UGC.

→ Para o problema, introduzimos uma nova variante da UGC, para o caso em que os grafos são bipartidos e os graus dos vértices em uma partição são os mesmos.

Conjectura 3 (Bipartite Unique Games Conjecture: BUGC)**.** *Dados quaisquer* $\varepsilon, \delta > 0$, *existe algum* $k > 0$ *dependente de* ε *e* δ, *tal que para o problema Unique Games com universo de tamanho k em grafos bipartidos nos quais todos os vértices de uma partição têm o mesmo grau, é* NP-*Difícil distinguir entre instâncias nas quais pelo menos uma fração de* $1-\varepsilon$ *das restrições pode ser satisfeita, e instâncias nas quais no máximo uma fração de* δ *das restrições pode ser satisfeita.*

- → Não provaremos a equivalência entre a BUGC e a UGC, assim como não o fizemos para a LUGC.

- → A partir dela, podemos enunciar o seguinte teorema, que é o principal resultado para provar o teorema 2:

> **Teorema 3** (BUGC-PCP). *Supondo a BUGC, para qualquer constante positiva $\gamma > 0$ e qualquer $\rho \in (-1, 0)$, NP $\subseteq PCP(\log(n), 2)$, onde o verificador tem completeness no mínimo $\frac{1}{2}(1-\rho) - \gamma$ e soundness no máximo $\frac{1}{\pi}\arccos\rho + \gamma$ e o verificador aceita apenas se dois bits não são iguais.*

Prova do Teorema 2.

- → Para qualquer instância de um problema NP-Completo Π e um verificador como o do Teorema 3, podemos criar uma instância $G(V_1, V_2, E)$ do Max-Cut equivalente ao CSP gerado:
  - Sejam os bits lidos da prova vértices de G, de forma que:
    - Se $v = 0$, então $v \in V_1$
    - Se $v = 1$, então $v \in V_2$
  - Para toda possível *string* de bits do verificador, cria-se uma aresta $e \in E$, tal que:
    - Se vértices comparados tem mesmo valor, então e não entra no corte
    - Se vértices comparados valores distintos, então e entra no corte
  - Por construção, o número de restrições satisfeitas no SCP representa o número de arestas no corte do grafo G.

- → Pelos conceitos de PCP, temos que:
  - Instâncias SIM de Π terão pelo menos uma fração c de restrições aceitas
  - Instâncias NÃO de Π terão no máximo uma fração s de restrições aceitas

- → Tais condições implicam que qualquer α-aproximação, tal que $\alpha > \frac{s}{c}$, resultariam em distinguir entre instâncias SIM e NÃO do problema π, NP-Completo, em tempo polinomial.

- → Como isso é válido para quaisquer $\gamma > 0$ e $\rho \in (-1, 0)$, temos que não existe α-aproximação para $\alpha > \min_{\rho \in (-1,0)} \frac{s}{c}$.

- → Enfim, vemos que $\min_{\rho \in (-1,0)} \frac{s}{c} = \min_{\rho \in (-1,1)} \frac{s}{c} = \min_{\rho \in (-1,1)} \frac{\frac{1}{\pi}\arccos\rho}{\frac{1}{2}(1-\rho)}$.

- → Assim, considerando o Teorema 3, o Teorema 2 é verdade: não há α-aproximação para o Max-Cut para $\alpha > .878$, a não ser que $\mathrm{P} = \mathrm{NP}$.

□

5.1 Construindo o Verificador PCP

→ Precisamos criar um algoritimo verificador de forma que as condições do Teorema 3 sejam satisfeitas.

→ Criaremos um algoritimo verificador e mostraremos que há uma prova π para uma instância SIM do BUGC em que o verificador aceita com probabilidade $\geq completeness = \frac{1}{2}(1-\rho)$

→ Soundness não será provada.

→ **Conceitos necessários:**

- ***Funções ditadoras:*** $f : \{0,1\}^k \rightarrow \{0,1\}$, tal que para $f(b_1, b_2, ..., b_k) = b_i$ tal que $i \in [0,k]$
- ***Ruído:*** Dado $x \in \{0,1\}^k$, podemos criar y invertendo independentemente cada bit de x com probabilidadade $\frac{1}{2}(1-\rho)$. Denota-se esse processo por $y \sim_\rho x$.
- ***Bits permutados pela aresta:*** Dado $x \in \{0,1\}^k$ e uma permutação $\pi : [k] \rightarrow [k]$, dada uma aresta (v,w), seja $x \circ \pi_{vw} = (x_{\pi vw(1)}, x_{\pi vw(2)}, ..., x_{\pi vw(k)})$.
- **Observações:**
  - A sensibilidade à ruído de uma função ditadora é equivalente a probabilidade de o bit ditador ser invertido

→ Note que dada a aresta (v,w) onde v tem rótulo i codificado por f_v e w tem rótulo j codificado por f_w, temos que $f_v(x) = f_w(x \circ \pi_{vw})$ para todo $x \in \{0,1\}^k$.

→ Queremos codificar uma prova para uma instância SIM do BUGC.
Dada uma rotulação como prova, codificaremos os rótulos de todos os vértices v em bits da seguinte forma:

- Cria-se uma função ditadora $f_v : \{0,1\}^k \rightarrow \{0,1\}$
- Para codificar que o vértice v possui rótulo i, temos que, para uma entrada $x \in \{0,1\}^k$, $f_v(x) = x_i$ tal que f_v é uma função ditadora e seu resultado é o i-ésimo bit de x.
- Para todas as possíveis *strings* de entrada $x \in \{0,1\}^k$, listamos os possíveis valores de $f_v(x)$ em um bloco de bits.
- A prova π será a concatenação de todos os blocos de bits gerados para cada um dos vértices codificados da prova.

→ **Verificador:**

- Dada uma instância $G(V_1, V_2, E)$ do BUG
- Seleciona-se $v \in V_1$ aleatoriamente
- Seleciona-se dois vizinhos $w, u \in V_2$ de v aleatoriamente e independentemente.
- Seleciona-se uma string aleatória $x \in \{0,1\}^k$ e gera-se $y \sim_\rho x$.
- Por fim, compara os dois bits de prova $f_w(x \circ \pi_{vw})$ e $f_u(y \circ \pi_{vu})$
- Retorna SIM se $f_w(x \circ \pi_{vw}) \neq f_u(y \circ \pi_{vu})$

5.2 Completeness

Lema 3. *Para qualquer $\rho \in [-1, 1]$, se pelo menos uma fração de $1-\varepsilon$ das arestas da instância do BUG são satisfeitas, então pode-se provar que o verificador aceita a prova com probabilidade de pelo menos $(1-2\varepsilon)\cdot\frac{1}{2}(1-\rho)$.*

Prova do Lema 3.

→ Probabilidade de que o verificador descrito com a prova π e uma instância SIM da BUGC (que tenha pelo menos uma fração $\frac{1}{2}(1-\rho)$ da arestas satisfatíveis) retorna SIM:

- Probabilidade de (v, w) seja satifatível é $\geq (1-\varepsilon)$
- Probabilidade de (v, u) seja satifatível é $\geq (1-\varepsilon)$
- Probabilidade de ambas (v, w) e (v, u) serem satisfeitas é $\geq (1-2\varepsilon)$
- Probabilidade de que $f_w(x \circ \pi_{vw}) = f_u(x \circ \pi_{vu})$ é $\geq (1-2\varepsilon)$
- Probabilidade de que $f_u(x \circ \pi_{vu}) \neq f_u(y \circ \pi_{vu})$ é $\frac{1}{2}(1-\rho)$
- Probabilidade de que $f_w(x \circ \pi_{vw}) = f_u(x \circ \pi_{vu})$ e $f_u(x \circ \pi_{vu}) \neq f_u(y \circ \pi_{vu})$ é equivalente a probabilidade de que $f_w(x \circ \pi_{vw}) \neq f_u(y \circ \pi_{vu})$
- Portanto, retorna SIM com probabilidade $\geq (1-2\varepsilon)\frac{1}{2}(1-\rho)$

→ Provamos a *completeness* do verificador. □

5.3 Soundness

Lema 4. *Para qualquer $\rho \in (-1, 0)$ e qualquer $\gamma > 0$, se pelo menos uma fração δ das arestas da instância do BUG forem satisfatíveis, então pode-se provar que o verificador aceita a instância com probabilidade de no máximo $\frac{1}{\pi}\arccos\rho + \gamma$*

Prova do Lema 4.

→ Parte difícil da prova da redução. Não será discutida aqui.

→ Pode ser encontrada em [2].

→ Aceitemos que o *soundness* do Teorema 3 vale:
Para uma instância NÃO do BUGC (ou seja, no máximo uma fração $s \leq \frac{1}{\pi}\arccos\rho$ das arestas são satisfeitas), temos que o verificador retorna SIM para qualquer prova π com probabilidade de no máximo s. □

5.4 Concluindo

→ Os Lemas 3 e 4 provam que podemos criar o verificador PCP do Teorema 3;

→ Temos que o Teorema 3 implica o Teorema 2, dada a redução do verificador para o Max-Cut;

→ **Concluindo, o Teorema 2 (inaproximabilidade do Max-Cut para $\alpha > .878$) é verdadeiro.**

6 2-2 Games Conjecture

→ Durante muito tempo, a comunidade acadêmica ficou dividida no que tange à veracidade da UGC. Ainda assim, é tópico de pesquisa há muitos anos, assim como suas variantes.

→ Uma variação do Unique Games é o *2-2 Games*. Nesse problema, em vez das *labels* serem únicas para uma restrição, existem *duas alternativas* que a satisfazem.

→ Em janeiro de 2018, um artigo foi publicado pelo Subhash Khot (com outros pesquisadores) que, unido com outros publicados recentemente, *prova* a 2-2 Games Conjecture, variante mais fraca da UGC para o 2-2 Games.

→ Isso convenceu muitos céticos (alguns dos maiores céticos!) de que a UGC deve ser verdadeira, pela proximidade em relação ao 2-2 GC.

→ É dito que isso prova *metade* da UGC.

→ Mais informações sobre a 2-2 Games Conjecture e suas consequências em [3].

Exercício

→ a) Dado a instância do problema do Unique Games a seguir, construa uma instância equivalente do problema Unique Label Cover (vértices, arestas e permutações):

Universo: U = {0, 1, 2, 3};

Variáveis: x_a, x_b, x_c, x_d;

Restrições:

$f_1(x_a, x_b) : f_1(0,2) = 1, f_1(1,3) = 1;$

$f_2(x_a, x_c) : f_2(0,2) = 1, f_2(1,3) = 1;$

$f_3(x_b, x_c) : f_3(0,1) = 1, f_3(2,3) = 1;$

$f_4(x_c, x_d) : f_4(0,3) = 1, f_4(1,2) = 1;$

→ b) Verifique se todas as arestas da instância são satisfatíveis.

→ c) Para essa instância pequena, é possível rapidamente verificar quantas arestas são satisfatíveis. Qual a consequência da UGC para o problema, com instâncias quaisquer?

Referências

[1] S. Arora, B. Barak, and D. Steurer. Subexponential algorithms for unique games and related problems. *J. ACM*, 62(5):42:1–42:25, Nov. 2015.

[2] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM J. Comput.*, 37(1):319–357, Apr. 2007.

[3] E. Klarreich. First big steps toward proving the unique games conjecture, Apr 2018.

[4] Subhash Khot. On the unique games conjecture. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 3–, Oct 2005.

[5] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2011.