Unique Games Conjecture

Algoritmos de Aproximação

Felipe L. De Mello, Victor F. Ferrari e Vinícius C. Espindola

14/11/2019

Instituto de Computação – Unicamp

Introdução

Motivação

Theorem 1.16: *Assuming the unique games conjecture, if there exists an α-approximation algorithm for the vertex cover problem with constant $\alpha < 2$, then $\mathrm{P} = \mathrm{NP}$.*

Theorem 5.32: *Assuming a variant of the unique games conjecture, for any constant $k > 3$, it is NP-hard to decide if a graph can be colored with only 3 colors, or needs at least k colors.*

Theorem 15.12: *Assuming the unique games conjecture, there is no α-approximation algorithm for the sparsest cut problem for constant α unless $\mathrm{P} = \mathrm{NP}$.*

Theorem 8.10: *Assuming the unique games conjecture, for any constant $\alpha \geq 1$, there is no α-approximation algorithm for the multicut problem unless $\mathrm{P} = \mathrm{NP}$.*

Motivação

Theorem 6.11: *Given the unique games conjecture there is no α-approximation algorithm for the maximum cut problem with constant*

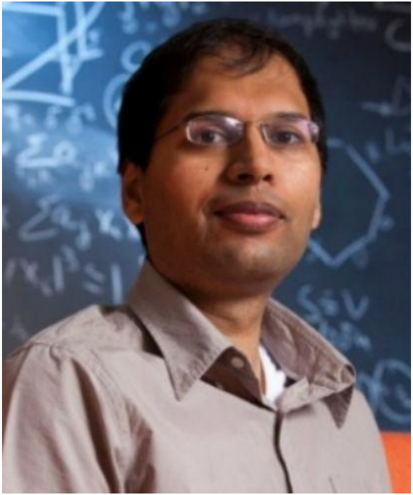
$$\alpha > \min_{-1 \le x \le 1} \frac{\frac{1}{\pi}\arccos(x)}{\frac{1}{2}(1-x)} \ge .878$$

unless $\mathrm{P} = \mathrm{NP}$.

Motivação

- 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?

Histórico



Subhash Khot

Histórico

- Coloração de grafos;
- Há diferença entre 3 cores e k cores?
- Mais em [2].

Unique Games e Unique Label Cover

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).

Unique Games

- 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.

Unique Games

- 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 (Unique Games Conjecture: UGC)

Dados quaisquer $\epsilon, \delta > 0$, 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 $1 - \epsilon$ 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.

- 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!

Unique Label Cover

- 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, $U = 1 \ldots k$.
  - Assim, podemos definir a permutação como $\pi(i) = j$ se $f(i, j) = 1$.

Transformação Unique Games-Unique Label Cover

Assim, a transformação é:

Transformação-UG-ULC

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

Unique Label Cover

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

Unique Label Cover

- Similarmente, saber se *nenhuma* aresta é satisfatível também é trivial.
- Porém, o mesmo não pode ser dito para todos os outros casos.
- 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.

Algoritmos para Unique Label Cover/Unique Games

- Existe um algoritmo de aproximação de fator $1 - \sqrt{\epsilon \log n}$ para o Unique Games/Unique Label Cover, baseado em *Semidefinite Programming*.
  - Esse algoritmo funciona para instâncias nas quais uma fração de $1 - \epsilon$ das arestas/restrições são satisfatíveis.
  - Se $\epsilon \in O(1/\log n)$, essa aproximação é constante.
  - Não iremos mostrar esse algoritmo hoje, mas está especificado e demonstrado em [4].
- Há também algoritmos *subexponenciais* para o problema e outros relacionados, ao contrário de diversos outros problemas NP-Difíceis. Alguns podem ser vistos em [1].

Consequências da UGC

Problemas Relacionados à UGC

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}	$\alpha_{MC}-\varepsilon$	$\frac{17}{16}-\varepsilon$	[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.

Consequências da UGC

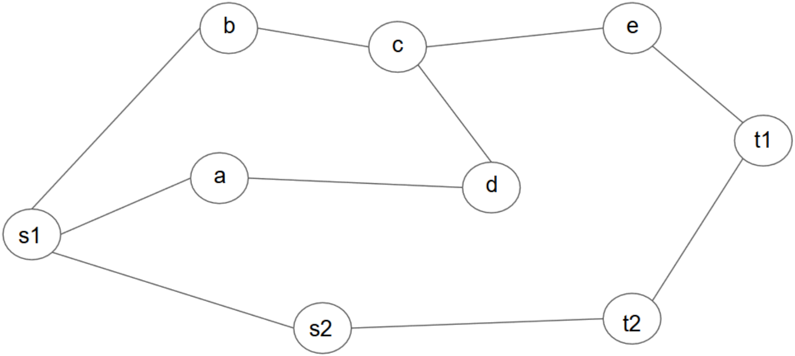
- Desde 2002, a UGC teve diversas aplicações em problemas relacionados.
- A figura 1 foi extraída de [3], que também possui diversos desenvolvimentos e conclusões em cima da UGC.
- Como visto na figura, o principal uso da UGC foi para provas de *inaproximabilidade*, muitas vezes junto com outra técnica, como PCP.
- Ela também deu origem a diversas *variantes*.
- Deste ponto em diante, veremos duas reduções para problemas que provam inaproximabilidade ou que uma aproximação conhecida é a melhor possível.

Redução para Multicut

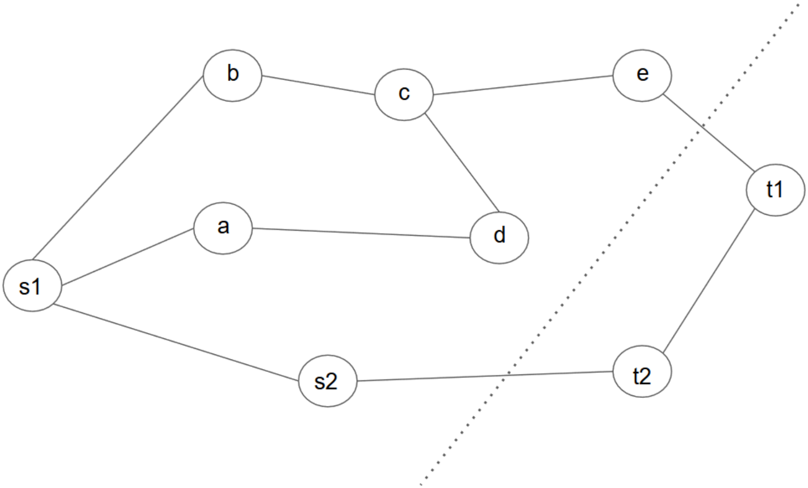
Multicorte

- 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 $s_1 - t_1, \ldots, s_k - t_k$, $s, t \in V$.
  - Solução: conjunto de arestas F nas quais ao serem removidas desconectam todos os pares $s_1 - t_1, \ldots, s_k - t_k$.
  - Objetivo: minimizar o custo total das arestas F removidas.

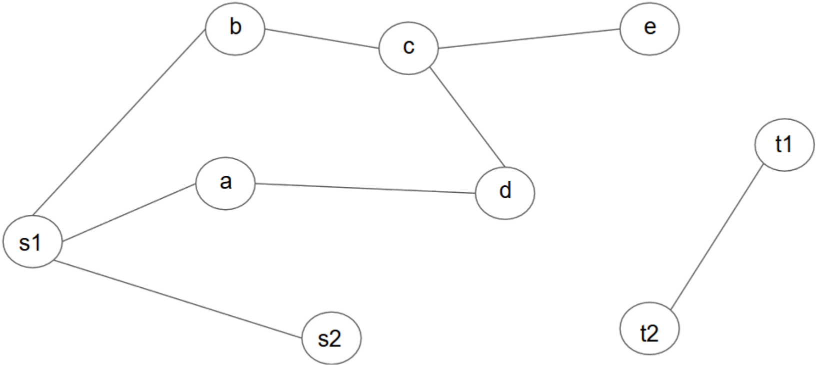
Exemplo de Multicorte



Exemplo de Multicorte



Exemplo de Multicorte



Redução para Multicorte

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).

Redução para Multicorte

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.

Redução para Multicorte

Conjectura (Linear Unique Games Conjecture: LUGC)

Dados quaisquer $\epsilon, \delta > 0$, 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 $1 - \epsilon$ das arestas pode ser satisfeita, e instâncias nas quais no máximo uma fração de δ das arestas pode ser satisfeita.

Redução para Multicorte

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

Lema (1)

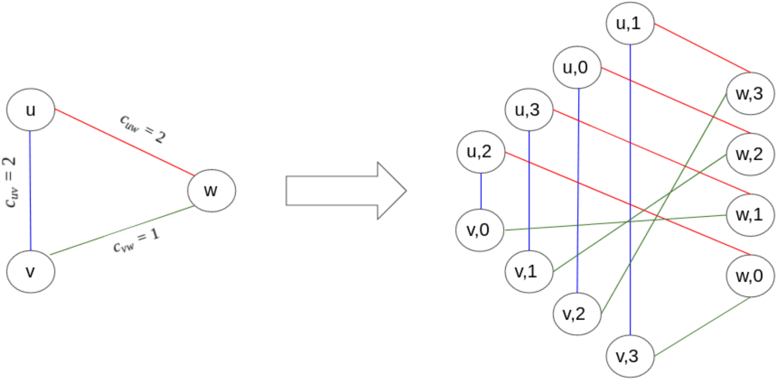
Para qualquer ϵ *tal que* $0 \leq \epsilon \leq 1$, *dado uma solução viável de uma instância de MAX 2LIN(k) que satisfaz pelo menos* $(1 - \epsilon)|\, E \,|$ *arestas, então existe uma solução viável para uma instância do multicut com custo de no máximo* $\epsilon|\, E' \,|$.

Redução para Multicorte

Prova do Lema 1:

- 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 $V' = k|\, V\,|$;

Redução para Multicorte

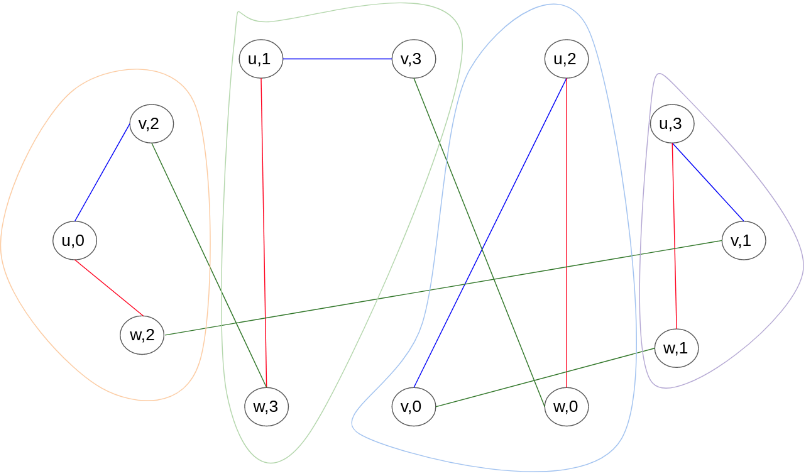


Redução para Multicorte

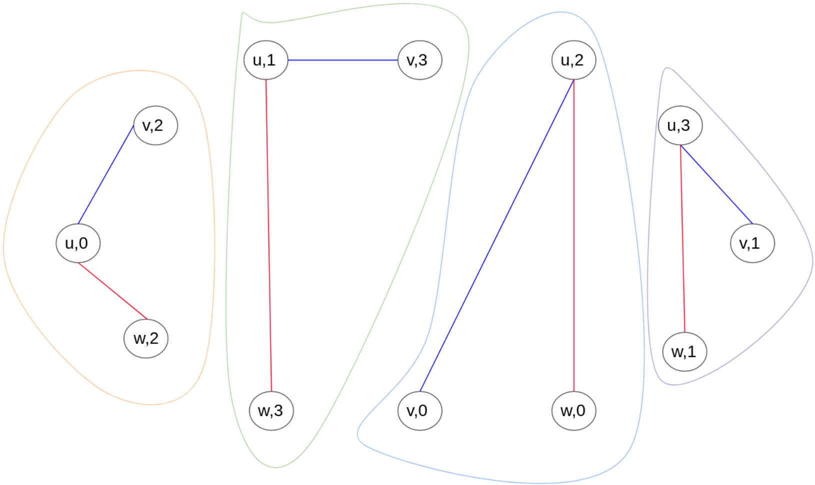
Solução do multicorte:

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

Redução para Multicorte



Redução para Multicorte



Redução para Multicorte

Custo dessa 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:

Redução para Multicorte

- 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 $\epsilon | E |$ arestas não são satisfeitas em MAX 2LIN(k), no máximo $\epsilon k | E | = \epsilon | E' |$ arestas são removidas no multicorte.

Lema (2)

Para qualquer ϵ tal que $0 \leq \epsilon \leq 1$, dado uma solução de uma instância do multicorte de custo máximo $\epsilon| E' |$, 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 [4].

Redução para Multicorte

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

Redução para Multicorte

Prova do Teorema 1:

- Se a instância do MAX LIN2(k) satisfaz no máximo $\delta| E |$ arestas, então este algoritmo satisfaz no máximo $\delta| E |$ arestas;
- Fazendo $\epsilon < \frac{1-\delta}{2\alpha}$, então temos que $(1 - 2\epsilon\alpha)| E | > \delta| E |$.
- 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 $\mathrm{P} = \mathrm{NP}$. $\square$

Redução para Max-Cut

O problema Max-Cut

- Relembrando o problema do Max-Cut:
  - Entrada:
    - grafo $G = (V, E)$ não-direcionado;
  - Solução:
    - Partição de vértices V_1 e V_2:
  - Objetivo:
    - Maximizar o número de arestas no corte
    - $\forall e \in E$, temos que e está no corte se: $e = (u, v) | u \in V_1 \;\&\; v \in V_2$

Inaproximibilidade do Max-Cut

Teorema

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 $\mathrm{P} = \mathrm{NP}$.

- Pelo teorema, é *UG-Difícil* aproximar o problema do Max-Cut por qualquer constante α maior que .878.
- Para a redução da UG para Max-Cut, utilizamos uma variação equivalente da UGC denominada *Bipartite Unique Games Conjecture*.

Inaproximibilidade do Max-Cut

Conjectura (BUGC)

Dados quaisquer $\epsilon, \delta > 0$, 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 $1 - \epsilon$ das restrições pode ser satisfeita, e instâncias nas quais no máximo uma fração de δ das restrições pode ser satisfeita.

- A probabilidade selecionar uma aresta aleatória (v, u) dado um vértice v, é a mesma para qualquer aresta
- Dado v, podemos sortear uma segunda aresta (v, w), resultando em duas arestas que compartilham v
- seleção aleatória e uniforme
- podemos selecionar ϵ arbitariamente pequeno

Relembrando PCP

- **Verificador PCP:** Recebe uma instância x de um problema NP e uma prova π. Lendo r bits aleatórios, acessa apenas q bits da prova.
  - **Completeness:** Para uma instância SIM, há uma prova com probabilidade de aceite de pelo menos c
  - **Soundness:** Para uma instância NÃO, qualquer prova tem probabilidade de aceite de no máximo s
- **CSPs:** Verificadores PCP são geram CSPs onde os bits acessados são as variáveis e os testes entre eles, restrições.

PCP e Max-Cut

Lema

Supondo a BUGC, para qualquer constante positiva $\gamma > 0$ e qualquer $\rho \in (-1, 0)$, $\mathrm{NP} \subseteq \mathit{PCP}(\log(n), 2)$, 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 são diferente.

- **Se o Lema é verdade, o Teorema também é?**
- Em outras palavras:
  O verificador gera um CSP equivalente ao Max-Cut?

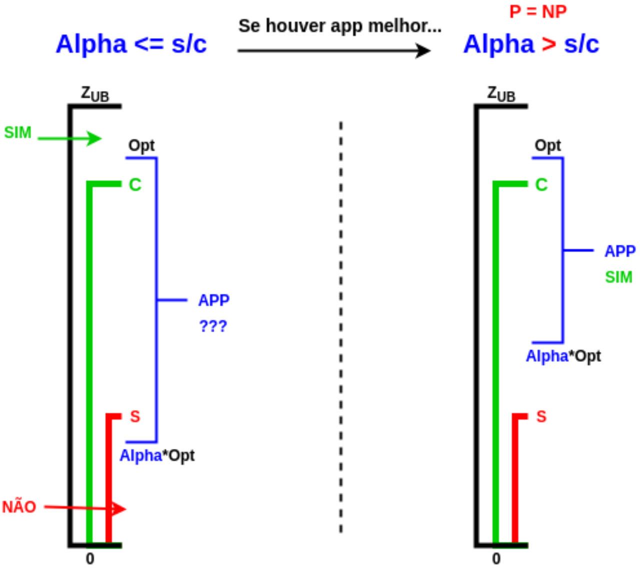
CSP e Max-Cut

- Dada uma instância de $\Pi \in$ NP-Completo codificada para o PCP mencionado
- Suponha que os bits da prova sejam vértices:
    - Se o bit for 1, $v \in V_1$
    - Se o bit for 0, $v \in V_2$
- Para todo possível par de bits (u, v) comparados, cria-se uma aresta (u, v) tal que $e \in E$
- Se os bits (u, v) forem diferentes, (u, v) está no corte.
- Geramos um resultado $G(V_1, V_2, E)$ do Max-Cut.

CSP e Max-Cut

- Suponha $P \neq NP$
- Temos $|E| = Z_{UB}$ restrições (número de arestas).
- Temos que Opt é o número restrições satisfeitas (corte).
- Suponha que há α-aproximação para o Max-Cut tal que $\alpha > \frac{s}{c}$
- podemos encontrar um valor aproximado X em tempo polinomial tal que $Opt \geq X > \frac{s}{c} \cdot Opt$
- Mas pelo teorema PCP:
  - SIM $\leftrightarrow Opt \geq c \cdot Z_{UB} \leftrightarrow Opt \geq X > s \cdot Z_{UB}$
  - NÃO $\leftrightarrow Opt \leq s \cdot Z_{UB} \leftrightarrow s \cdot Z_{UB} \geq X > \frac{s}{c} \cdot Opt$
- Logo, podemos resolver em tempo polinomial um problema NP-Completo
- **Contradição:** Não há α-app. a não ser que $P = NP$

CSP e Max-Cut

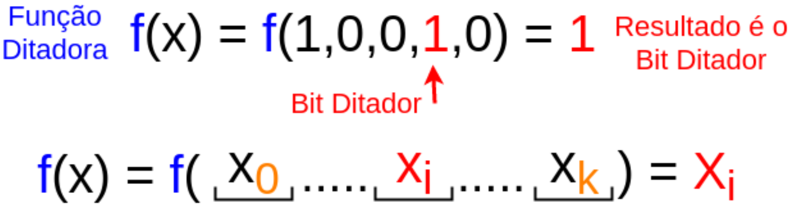


P = NP
Alpha <= s/c
Se houver app melhor...
Alpha > s/c
Z_UB
SIM
Opt
C
APP
???
S
Alpha*Opt
NÃO
0
Z_UB
Opt
C
APP
SIM
Alpha*Opt
S
0

CSP e Max-Cut

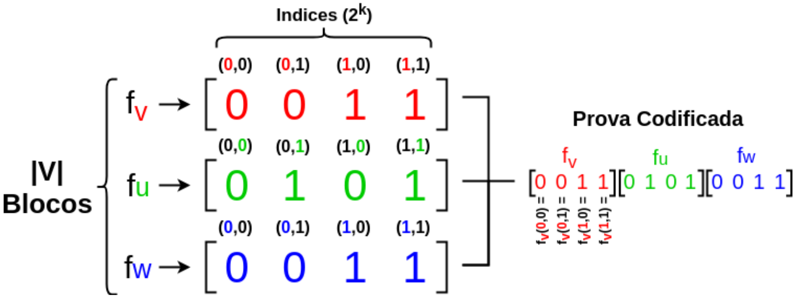
- **Se o Lema é verdade, o Teorema também é? SIM**
- Mas como construir o PCP descrito pelo Lema?
- Se a conjetura BUGC for verdadeira, isso é possível!
- **Parte difícil:** Codificar uma instância da BUGC em um PCP e provar que lendo apenas 2 bits:
  - Há uma prova para caso SIM aceita com probabilidade $\geq c$
  - Toda prova caso NÃO é aceita com probabilidade $\leq s$
- Lembrando: $c \geq \frac{1}{2}(1-\rho)$

- **Funções ditadoras:** $f : \{0,1\}^k \rightarrow \{0,1\}$, tal que para $f(x_1, ..., x_i, ..., x_k) = x_i$ para algum $i \in [0, k]$



Codificando a prova

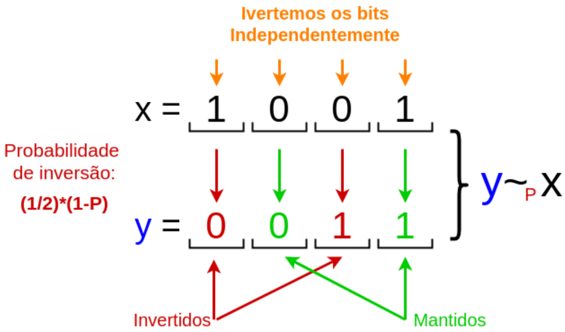
- Suponha 2 rótulos possíveis: $L = \{1, 2\}$ & $|L| = k = 2$
- Dado a prova: $v = 1$, $u = 2$ e $w = 1$ (Vértice=Rótulo)
- Temos as funções ditadoras:
  $x \in \{0,1\}^k,\ f_v(x_1, -) = x_1,\ f_u(-, x_2) = x_2,\ f_w(x_1, -) = x_1$



- Prova codificada: $[0,0,1,1][0,1,0,1][0,0,1,1]$ (bit vector)
- Tamanho: $2^k \cdot 2^k \cdot 2^k = |V| \cdot 2^k$ (*long Code*)

Conceitos necessários

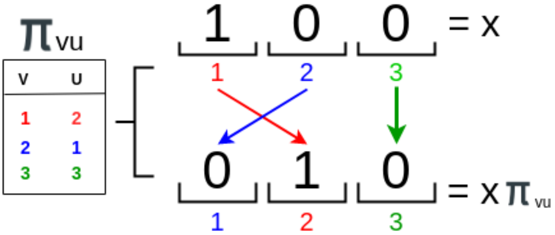
- **Ruído:** Dado $x \in \{0,1\}^k$, podemos criar y invertendo independentemente cada bit de x com probabilidadade $\frac{1}{2}(1-\rho)$. Denota-se esse processo por $y \sim_\rho x$



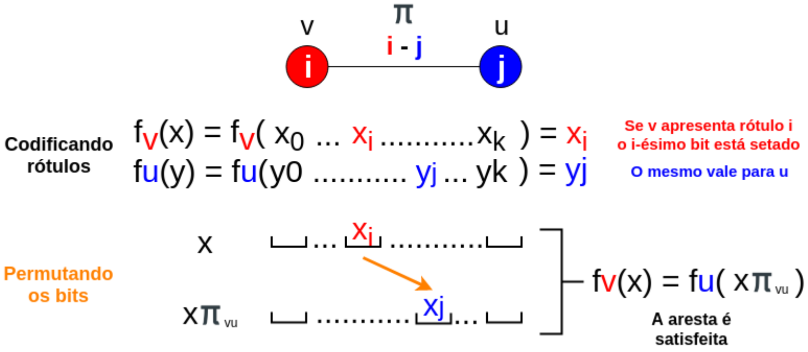
- **OBS:** Probabilidade de alterar o resultado de uma função ditadora é a probabilidade de inversão do bit ditador

Conceitos necessários

- **Permutação de bits** - Suponha tal permutação:



- Note que dada uma aresta $(v, u) \in E$ tal que v tem rótulo i e que u tem rótulo j, e dado um $x \in \{0,1\}^k$ qualquer, temos:
  *** $\pi_{vu}(i) = j \rightarrow f_v(x) = f_u(x \circ \pi_{vu})$
  uma vez que o i-ésimo bit de x é levado para o j-ésimo bit de $x \circ \pi_{vu}$ pela permutação



O verificador PCP

- Paramêtros:
  - $G(V_1, V_2, E)$ - Instância do BUGC
  - π - Prova codificada
- **Algoritmo:**
  - Seleciona um vértice $v \in V_1$
  - Seleciona um vizinho de v, $u \in V_2$
  - Seleciona outro vizinho de v, $w \in V_2$
  - Sorteia uma *string* $x \in \{0, 1\}^k$
  - Calcula $x \circ \pi_{vu}$
  - Calcula $y \sim_\rho x$
  - Calcula $y \circ \pi_{vw}$
  - **Query:** $b_u \leftarrow f_u(x \circ \pi_{vu})$; $b_w \leftarrow f_w(y \circ \pi_{vw})$
  - Se $b_u \neq b_w$, retorna SIM, caso contrário, NÃO.
- **Se $G(V_1, V_2, E)$ tem pelo menos $\geq 1 - \epsilon$ das arestas satisfatíveis, qual a probabilidade de SIM?**
- **Lembrando:** Precisa ser $\geq \frac{1}{2}(1 - \rho)$ (*completeness*)

O verificador PCP

- **Duas arestas aleatórias:**
    - Seleciona um vértice $v \in V_1$
    - Seleciona um vizinho de v, $u \in V_2$
    - Seleciona outro vizinho de v, $w \in V_2$
- Todo vértice $v \in V_1$ tem mesmo grau (BUGC)
- (v, u) e (v, w) - aleatoriedade e uniformidade
- Probabilidade de uma aresta ser satisfeita: $\geq (1 - \epsilon)$
- Então, a probabilidade de (v, u) e (v, w) serem satisfeitas é: $(1 - 2 \cdot \epsilon) \leq (1 - 2\epsilon + \epsilon^2) = (1 - \epsilon)^2$

O verificador PCP

- **Inserção de ruído:**
  - Calcula $y \sim_\rho x$
  - Calcula $y \circ \pi_{vw}$
- Qual a probabilidade de que $f_w(y \circ \pi_{vw}) \neq f_w(x \circ \pi_{vw})$?
- Probabilidade de **inversão do bit ditador**: $\frac{1}{2}(1-\rho)$

O verificador PCP

- **Acessando a prova:**
  - **Query:** $b_u \leftarrow f_u(x \circ \pi_{vu})$; $b_w \leftarrow f_w(y \circ \pi_{vw})$
- $f_u()$ e $f_w()$ são os blocos codificados
- $x \circ \pi_{vu}$ e $y \circ \pi_{vw}$ são índices dos blocos
- $f_u(x \circ \pi_{vu})$ e $f_w(y \circ \pi_{vw})$ representam 2 bits

O verificador PCP

- **Comparando bits:**
  - Se $b_u \neq b_w$, retorna SIM, caso contrário, NÃO.
- **Probabilidade de que $f_u(x \circ \pi_{vu}) \neq f_w(y \circ \pi_{vw})$?**
- Se (v, u) satisfeita, $f_v(x) = f_u(x \circ \pi_{vu}) \rightarrow (1 - \epsilon)$
- Se (v, w) satisfeita, $f_v(x) = f_w(x \circ \pi_{vw}) \rightarrow (1 - \epsilon)$
- $f_u(x \circ \pi_{vu}) = f_w(x \circ \pi_{vw}) \rightarrow \geq (1 - 2 \cdot \epsilon)$
- $f_w(x \circ \pi_{vw}) \neq f_w(y \circ \pi_{vw}) \rightarrow \frac{1}{2}(1 - \rho)$
- Probabilidade de ambas satisfeitas e inverteida com ruído?
  $\geq (1 - 2 \cdot \epsilon) \cdot \frac{1}{2}(1 - \rho)$
- Podemos escolher ϵ arbritariamente pequeno:
  $\geq \frac{1}{2}(1 - \rho)$
- *Completeness* provado

Soundness

- **E o soundness?**
- Parte realmente difícil da prova
- Fica pra próxima
- Aceitemos que $s \leq \frac{1}{\pi}(\arccos \rho)$

Concluindo

- Podemos construir o PCP com as condições dadas:

Lema

Supondo a BUGC, para qualquer constante positiva $\gamma > 0$ *e qualquer* $\rho \in (-1, 0)$, $\mathrm{NP} \subseteq PCP(\log(n), 2)$, *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.*

Concluindo

- Se o lema é verdade, mostramos que o teorema também é:

Teorema

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 $\mathrm{P} = \mathrm{NP}$.

OBS: Podemos mostrar que:

$$\min_{\rho \in (-1,0)} \frac{\frac{1}{\pi} \arccos x}{\frac{1}{2}(1-x)} = \min_{\rho \in [-1,1]} \frac{\frac{1}{\pi} \arccos x}{\frac{1}{2}(1-x)}$$

Processo comum: Rotulação codificada por funções ditadoras no estilo *long code* para reduzir ao problema de interesse

2-2 Games Conjecture

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.

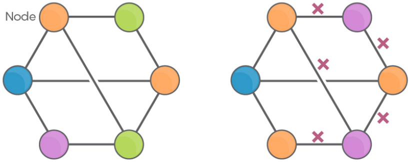
What Is the 2-2 Games Conjecture?

A recent proof of this conjecture, which allows two color choices instead of one, brings researchers closer to proving the Unique Games Conjecture.

THE GAME: Color nodes in a network according to predetermined constraints.

REVISED CONSTRAINTS (simplified to just two for the purposes of this graphic):

1. Joined nodes cannot be the same color.
2. If a node is orange then its adjoining nodes must be either blue or green.



✔ Constraints satisfied.

✘ Constraints not satisfied.

THE PROOF tells us (roughly speaking) that, given these broader constraints, there is no efficient algorithm capable of identifying colorings, for every conceivable network, that satisfy even a tiny fraction of the number of constraints the optimal coloring satisfies.

Fonte: [2].

Consequências da Prova da 2-2 Games Conjecture

- 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 [2].

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?

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.

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

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

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