Algoritmos de Aproximação

Segundo Semestre de 2012

Problema de decisão

Problema de decisão:

conjunto I de instâncias e

função $f : I \to \{\text{SIM}, \text{NÃO}\}$

Problema de decisão

Problema de decisão:
conjunto I de instâncias e
função $f : I \to \{\text{SIM}, \text{NÃO}\}$

Exemplo: Problema circuito hamiltoniano.

I: conjunto de todos os grafos.

Para todo G em I,

$$f(G) = \begin{cases} \text{SIM} & \text{se } G \text{ tem um circuito hamiltoniano} \\ \text{NÃO} & \text{caso contrário} \end{cases}$$

Problema de decisão

Problema de decisão:
conjunto I de instâncias e
função $f : I \to \{\text{SIM}, \text{NÃO}\}$

Exemplo: Problema circuito hamiltoniano.

I: conjunto de todos os grafos.

Para todo G em I,

$$f(G) = \begin{cases} \text{SIM} & \text{se } G \text{ tem um circuito hamiltoniano} \\ \text{NÃO} & \text{caso contrário} \end{cases}$$

O valor de $f(X)$ é
a resposta do problema para a instância X em I.

Modelo de computação

Algoritmo:
programa escrito em uma linguagem como `C` ou `Pascal`.

Modelo de computação

Algoritmo:
programa escrito em uma linguagem como `C` ou `Pascal`.

Custo de uma operação:
proporcional ao número de bits dos operandos

Custo de um acesso à memória:
proporcional ao número de bits dos operandos.

Modelo de computação

Algoritmo:
programa escrito em uma linguagem como `C` ou `Pascal`.

Custo de uma operação:
proporcional ao número de bits dos operandos

Custo de um acesso à memória:
proporcional ao número de bits dos operandos.

(Isto é polinomialmente equivalente à máquina de Turing.)

Modelo de computação

Algoritmo:
programa escrito em uma linguagem como `C` ou `Pascal`.

Custo de uma operação:
proporcional ao número de bits dos operandos

Custo de um acesso à memória:
proporcional ao número de bits dos operandos.

(Isto é polinomialmente equivalente à máquina de Turing.)

Um algoritmo resolve ou decide um problema de decisão (I, f) se, para cada X em I, o algoritmo encontra a resposta para X, isto é, o algoritmo calcula $f(X)$.

A classe P

Um algoritmo resolve (ou decide) um problema de decisão (I, f) em tempo $\mathrm{O}(T(n))$ se, para cada n em N e cada X em I com $|X| = n$, o algoritmo encontra a resposta $f(X)$ para X em tempo $\mathrm{O}(T(n))$.

A classe P

Um algoritmo resolve (ou decide) um problema de decisão (I, f) em tempo $\mathrm{O}(T(n))$ se, para cada n em N e cada X em I com $|X| = n$, o algoritmo encontra a resposta $f(X)$ para X em tempo $\mathrm{O}(T(n))$.

Um problema de decisão é solúvel em tempo polinomial se existe algum k para o qual existe um algoritmo que resolve o problema em tempo $\mathrm{O}(n^k)$.

A classe P

Um algoritmo resolve (ou decide) um problema de decisão (I, f) em tempo $\mathrm{O}(T(n))$ se, para cada n em N e cada X em I com $|X| = n$, o algoritmo encontra a resposta $f(X)$ para X em tempo $\mathrm{O}(T(n))$.

Um problema de decisão é solúvel em tempo polinomial se existe algum k para o qual existe um algoritmo que resolve o problema em tempo $\mathrm{O}(n^k)$.

Tais problemas são ditos tratáveis.

A classe P

Um algoritmo resolve (ou decide) um problema de decisão (I, f) em tempo $\mathrm{O}(T(n))$ se, para cada n em N e cada X em I com $|X| = n$, o algoritmo encontra a resposta $f(X)$ para X em tempo $\mathrm{O}(T(n))$.

Um problema de decisão é solúvel em tempo polinomial se existe algum k para o qual existe um algoritmo que resolve o problema em tempo $\mathrm{O}(n^k)$.

Tais problemas são ditos tratáveis.

Classe de complexidade P:
conjunto dos problemas de decisão tratáveis
(isto é, que são solúveis em tempo polinomial)

A classe NP

Considere o problema circuito hamiltoniano.

A classe NP

Considere o problema circuito hamiltoniano.

Se a resposta para uma instância G for SIM,
então existe um circuito hamiltoniano C no grafo.

A classe NP

Considere o problema circuito hamiltoniano.

Se a resposta para uma instância G for SIM,
então existe um circuito hamiltoniano C no grafo.

Dados G e C, é possível verificar em tempo polinomial em $|G|$ se C é de fato um circuito hamiltoniano em G ou não.

A classe NP

Considere o problema circuito hamiltoniano.

Se a resposta para uma instância G for SIM,
então existe um circuito hamiltoniano C no grafo.

Dados G e C, é possível verificar em tempo polinomial em $|G|$ se C é de fato um circuito hamiltoniano em G ou não.

Ou seja, temos como certificar eficientemente que a resposta SIM está correta.

A classe NP

Considere o problema circuito hamiltoniano.

Se a resposta para uma instância G for SIM,
então existe um circuito hamiltoniano C no grafo.

Dados G e C, é possível verificar em tempo polinomial em $|G|$ se C é de fato um circuito hamiltoniano em G ou não.

Ou seja, temos como certificar eficientemente que a resposta SIM está correta.

Conseguimos certificar eficientemente a resposta NÃO?
Surpreendentemente, não se conhece nenhum método de certificação eficiente para a resposta NÃO no caso do problema circuito hamiltoniano.

A classe NP

Classe de complexidade NP: problemas para os quais a resposta SIM pode ser certificada e verificada em tempo polinomial.

A classe NP

Classe de complexidade NP: problemas para os quais a resposta SIM pode ser certificada e verificada em tempo polinomial.

Mais precisamente, um problema de decisão está em NP se existe um algoritmo A tal que

1. para qualquer instância X do problema com resposta SIM, existe Y em Σ^* tq $A(X, Y)$ devolve SIM;
2. para qualquer instância X do problema com resposta NÃO, para todo Y em Σ^*, $A(X, Y)$ devolve NÃO;
3. A consome tempo polinomial em $|X|$.

A classe NP

Classe de complexidade NP: problemas para os quais a resposta SIM pode ser certificada e verificada em tempo polinomial.

Mais precisamente, um problema de decisão está em NP se existe um algoritmo A tal que

1. para qualquer instância X do problema com resposta SIM, existe Y em Σ^* tq $A(X, Y)$ devolve SIM;
2. para qualquer instância X do problema com resposta NÃO, para todo Y em Σ^*, $A(X, Y)$ devolve NÃO;
3. A consome tempo polinomial em $|X|$.

Y é chamado de certificado para SIM da instância X.

certificado $=$ prova

Definição alternativa de NP

IDEIA

Verificador:
algoritmo aleatorizado, que olha apenas alguns bits da prova escolhidos aleatoriamente. Faz alguns cálculos e, dependendo do resultado, aceita a prova ou não.

Definição alternativa de NP

IDEIA

Verificador:
algoritmo aleatorizado, que olha apenas alguns bits da prova escolhidos aleatoriamente. Faz alguns cálculos e, dependendo do resultado, aceita a prova ou não.

Para qq instância X do problema com resposta SIM, existe $Y \in \Sigma^*$ tq $V(X, Y)$ devolve SIM com alta probabilidade;

Definição alternativa de NP

IDEIA

Verificador:
algoritmo aleatorizado, que olha apenas alguns bits da prova escolhidos aleatoriamente. Faz alguns cálculos e, dependendo do resultado, aceita a prova ou não.

Para qq instância X do problema com resposta SIM, existe $Y \in \Sigma^*$ tq $V(X, Y)$ devolve SIM com alta probabilidade;

Para qq instância X do problema com resposta NÃO, para todo $Y \in \Sigma^*$, $V(X, Y)$ devolve NÃO com boa probabilidade;

Definição formal

Instância X codificada em n bits.

Verificador usa $r(n)$ bits aleatórios para escolher $q(n)$ bits aleatórios de Y.

Definição formal

Instância X codificada em n bits.

Verificador usa $r(n)$ bits aleatórios para escolher $q(n)$ bits aleatórios de Y.

Verificador escolhe uma função $f : \{0,1\}^{q(n)} \rightarrow \{0,1\}$, e calcula f nos $q(n)$ bits de Y.

Aceita se f tem valor 1, senão rejeita Y.

Definição formal

Instância X codificada em n bits.

Verificador usa $r(n)$ bits aleatórios para escolher $q(n)$ bits aleatórios de Y.

Verificador escolhe uma função $f : \{0,1\}^{q(n)} \rightarrow \{0,1\}$, e calcula f nos $q(n)$ bits de Y.

Aceita se f tem valor 1, senão rejeita Y.

Se a resposta para X é SIM, existe Y tq o verificador aceita Y com probabilidade pelo menos c.
(parâmetro c é a completude do verificador)

Definição formal

Instância X codificada em n bits.

Verificador usa $r(n)$ bits aleatórios para escolher $q(n)$ bits aleatórios de Y.

Verificador escolhe uma função $f : \{0,1\}^{q(n)} \rightarrow \{0,1\}$, e calcula f nos $q(n)$ bits de Y.

Aceita se f tem valor 1, senão rejeita Y.

Se a resposta para X é SIM, existe Y tq o verificador aceita Y com probabilidade pelo menos c.
(parâmetro c é a completude do verificador)

Se a resposta para X é NÃO, para todo Y, o verificador aceita Y com probabilidade no máximo $s < c$.
(parâmetro s é a robustez do verificador)

Classes PCP

Instância X codificada em n bits.

Verificador usa $r(n)$ bits aleatórios para escolher $q(n)$ bits aleatórios de Y.

Verificador escolhe uma função $f : \{0,1\}^{q(n)} \rightarrow \{0,1\}$, e calcula f nos $q(n)$ bits de Y.

Aceita se f tem valor 1, senão rejeita Y.

Se a resposta para X é SIM, existe Y tq o verificador aceita Y com probabilidade pelo menos c.

Se a resposta para X é NÃO, para todo Y, o verificador aceita Y com probabilidade no máximo $s < c$.

$\mathrm{PCP}_{c,s}(r(n), q(n))$: problemas de decisão para os quais existe um tal verificador.

Classes PCP

$\mathrm{PCP}_{c,s}(r(n), q(n))$: problemas de decisão para os quais existe um tal verificador.

Classes PCP

$\mathrm{PCP}_{c,s}(r(n), q(n))$: problemas de decisão para os quais existe um tal verificador.

Da definição original de NP: para todo problema Π em NP, existe um polinômio p tq $\Pi \in \mathrm{PCP}_{1,0}(0, p(n))$.

Classes PCP

$\mathrm{PCP}_{c,s}(r(n), q(n))$: problemas de decisão para os quais existe um tal verificador.

Da definição original de NP: para todo problema Π em NP, existe um polinômio p tq $\Pi \in \mathrm{PCP}_{1,0}(0, p(n))$.

Teorema do PCP: Existe uma constante positiva k tq NP $\subseteq \mathrm{PCP}_{1,1/2}(\mathrm{O}(\lg n), k)$.

Classes PCP

$\mathrm{PCP}_{c,s}(r(n), q(n))$: problemas de decisão para os quais existe um tal verificador.

Da definição original de NP: para todo problema Π em NP, existe um polinômio p tq $\Pi \in \mathrm{PCP}_{1,0}(0, p(n))$.

Teorema do PCP: Existe uma constante positiva k tq NP $\subseteq \mathrm{PCP}_{1,1/2}(\mathrm{O}(\lg n), k)$.

Discussão: na aula.

- $2^{c \lg n} = n^c$ bits da prova;
- 2^{2^k} funções f distintas em k bits.

Constraint satisfaction problem

n variáveis $x_1, \ldots, x_n$

m restrições sobre subconjuntos das n variáveis

Constraint satisfaction problem

n variáveis $x_1, \ldots, x_n$

m restrições sobre subconjuntos das n variáveis

Objetivo: encontrar atribuição de valores para as variáveis que satisfaça o maior número das restrições.

Constraint satisfaction problem

n variáveis $x_1, \ldots, x_n$

m restrições sobre subconjuntos das n variáveis

Objetivo: encontrar atribuição de valores para as variáveis que satisfaça o maior número das restrições.

Existe a versão com peso nas restrições também.

Constraint satisfaction problem

n variáveis $x_1, \ldots, x_n$

m restrições sobre subconjuntos das n variáveis

Objetivo: encontrar atribuição de valores para as variáveis que satisfaça o maior número das restrições.

Existe a versão com peso nas restrições também.

Casos especiais: MAX CUT, MAX SAT.
(Restrição é satisfeita se resulta em valor 1.)

Constraint satisfaction problem

n variáveis $x_1, \ldots, x_n$

m restrições sobre subconjuntos das n variáveis

Objetivo: encontrar atribuição de valores para as variáveis que satisfaça o maior número das restrições.

Existe a versão com peso nas restrições também.

Casos especiais: MAX CUT, MAX SAT.
(Restrição é satisfeita se resulta em valor 1.)

Corolário do Teorema do PCP: Não existe α-aproximação para este problema com $\alpha > 1/2$, a menos que P $=$ NP.

(Ideia da prova na aula.)

Variantes do PCP

Teorema: Para quaisquer constantes ϵ, $\delta > 0$,
NP $\subseteq \mathrm{PCP}_{1-\epsilon,1/2+\delta}(\mathrm{O}(\lg n), 3)$, onde o verificador só pode usar as funções par ou ímpar.

Variantes do PCP

Teorema: Para quaisquer constantes ϵ, $\delta > 0$,
NP $\subseteq \mathrm{PCP}_{1-\epsilon,1/2+\delta}(\mathrm{O}(\lg n), 3)$, onde o verificador só pode usar as funções par ou ímpar.

par$(x_i, x_j, x_k) = 1$ sse $x_i + x_j + x_k$ é par
ímpar$(x_i, x_j, x_k) = 1$ sse $x_i + x_j + x_k$ é ímpar

Variantes do PCP

Teorema: Para quaisquer constantes ϵ, $\delta > 0$,
NP $\subseteq \mathrm{PCP}_{1-\epsilon,1/2+\delta}(\mathrm{O}(\lg n), 3)$, onde o verificador só pode usar as funções par ou ímpar.

par$(x_i, x_j, x_k) = 1$ sse $x_i + x_j + x_k$ é par
ímpar$(x_i, x_j, x_k) = 1$ sse $x_i + x_j + x_k$ é ímpar

Odd/even constraint satisfaction problem
Restrições só do tipo par(x_i, x_j, x_k) e/ou ímpar(x_i, x_j, x_k).

Variantes do PCP

Teorema: Para quaisquer constantes ϵ, $\delta > 0$,
NP $\subseteq \mathrm{PCP}_{1-\epsilon,1/2+\delta}(\mathrm{O}(\lg n), 3)$, onde o verificador só pode usar as funções par ou ímpar.

par$(x_i, x_j, x_k) = 1$ sse $x_i + x_j + x_k$ é par
ímpar$(x_i, x_j, x_k) = 1$ sse $x_i + x_j + x_k$ é ímpar

Odd/even constraint satisfaction problem
Restrições só do tipo par(x_i, x_j, x_k) e/ou ímpar(x_i, x_j, x_k).

Corolário: Não existe α-aproximação para este problema com $\alpha > 1/2$, a menos que P $=$ NP.

Variantes do PCP

Teorema: Para quaisquer constantes ϵ, $\delta > 0$,
NP $\subseteq \mathrm{PCP}_{1-\epsilon,1/2+\delta}(\mathrm{O}(\lg n), 3)$, onde o verificador só pode usar as funções par ou ímpar.

par$(x_i, x_j, x_k) = 1$ sse $x_i + x_j + x_k$ é par
ímpar$(x_i, x_j, x_k) = 1$ sse $x_i + x_j + x_k$ é ímpar

Odd/even constraint satisfaction problem
Restrições só do tipo par(x_i, x_j, x_k) e/ou ímpar(x_i, x_j, x_k).

Corolário: Não existe α-aproximação para este problema com $\alpha > 1/2$, a menos que P $=$ NP.

Uma $1/2$-aproximação é trivial.
(razão melhor possível, a menos que P $=$ NP)

L-redução para MAX E3SAT

Cada restrição ímpar(x_1, x_2, x_3) corresponde às cláusulas

$x_1 \vee x_2 \vee x_3$, $\bar{x}_1 \vee \bar{x}_2 \vee x_3$, $\bar{x}_1 \vee x_2 \vee \bar{x}_3$, $x_1 \vee \bar{x}_2 \vee \bar{x}_3$.

L-redução para MAX E3SAT

Cada restrição ímpar(x_1, x_2, x_3) corresponde às cláusulas
$x_1 \vee x_2 \vee x_3$, $\bar{x}_1 \vee \bar{x}_2 \vee x_3$, $\bar{x}_1 \vee x_2 \vee \bar{x}_3$, $x_1 \vee \bar{x}_2 \vee \bar{x}_3$.

Se ímpar$(x_1, x_2, x_3) = 1$,
4 cláusulas satisfeitas, senão apenas 3.

Semelhante para par.

L-redução para MAX E3SAT

Cada restrição ímpar(x_1, x_2, x_3) corresponde às cláusulas
$x_1 \vee x_2 \vee x_3$, $\bar{x}_1 \vee \bar{x}_2 \vee x_3$, $\bar{x}_1 \vee x_2 \vee \bar{x}_3$, $x_1 \vee \bar{x}_2 \vee \bar{x}_3$.

Se ímpar$(x_1, x_2, x_3) = 1$,
4 cláusulas satisfeitas, senão apenas 3.

Semelhante para par.

Análise da L-redução apresentada na aula.

L-redução para MAX E3SAT

Cada restrição ímpar(x_1, x_2, x_3) corresponde às cláusulas
$x_1 \vee x_2 \vee x_3$, $\bar{x}_1 \vee \bar{x}_2 \vee x_3$, $\bar{x}_1 \vee x_2 \vee \bar{x}_3$, $x_1 \vee \bar{x}_2 \vee \bar{x}_3$.

Se ímpar$(x_1, x_2, x_3) = 1$,
4 cláusulas satisfeitas, senão apenas 3.

Semelhante para par.

Análise da L-redução apresentada na aula.

Para problemas de maximização,

$$v \geq \mathrm{OPT}(I)(1 - ab(1 - \alpha)).$$

L-redução para MAX E3SAT

Cada restrição ímpar(x_1, x_2, x_3) corresponde às cláusulas
$x_1 \vee x_2 \vee x_3$, $\bar{x}_1 \vee \bar{x}_2 \vee x_3$, $\bar{x}_1 \vee x_2 \vee \bar{x}_3$, $x_1 \vee \bar{x}_2 \vee \bar{x}_3$.

Se ímpar$(x_1, x_2, x_3) = 1$,
4 cláusulas satisfeitas, senão apenas 3.

Semelhante para par.

Análise da L-redução apresentada na aula.

Para problemas de maximização,

$$v \geq \mathrm{OPT}(I)(1 - ab(1 - \alpha)).$$

Para $a = 7$ e $b = 1$, obtemos razão $7\alpha - 6$.

L-redução para MAX E3SAT

Cada restrição ímpar(x_1, x_2, x_3) corresponde às cláusulas
$x_1 \vee x_2 \vee x_3$, $\bar{x}_1 \vee \bar{x}_2 \vee x_3$, $\bar{x}_1 \vee x_2 \vee \bar{x}_3$, $x_1 \vee \bar{x}_2 \vee \bar{x}_3$.

Se ímpar$(x_1, x_2, x_3) = 1$,
4 cláusulas satisfeitas, senão apenas 3.

Semelhante para par.

Análise da L-redução apresentada na aula.

Para problemas de maximização,

$$v \geq \mathrm{OPT}(I)(1 - ab(1 - \alpha)).$$

Para $a = 7$ e $b = 1$, obtemos razão $7\alpha - 6$.

$7\alpha - 6 > \frac{1}{2}$ implica $\alpha > \frac{13}{14}$, donde...

L-redução para MAX E3SAT

Cada restrição ímpar(x_1, x_2, x_3) corresponde às cláusulas
$x_1 \vee x_2 \vee x_3$, $\bar{x}_1 \vee \bar{x}_2 \vee x_3$, $\bar{x}_1 \vee x_2 \vee \bar{x}_3$, $x_1 \vee \bar{x}_2 \vee \bar{x}_3$.

Se ímpar$(x_1, x_2, x_3) = 1$,
4 cláusulas satisfeitas, senão apenas 3.

Semelhante para par.

Análise da L-redução apresentada na aula.

L-redução para MAX E3SAT

Cada restrição ímpar(x_1, x_2, x_3) corresponde às cláusulas
$x_1 \vee x_2 \vee x_3$, $\bar{x}_1 \vee \bar{x}_2 \vee x_3$, $\bar{x}_1 \vee x_2 \vee \bar{x}_3$, $x_1 \vee \bar{x}_2 \vee \bar{x}_3$.

Se ímpar$(x_1, x_2, x_3) = 1$,
4 cláusulas satisfeitas, senão apenas 3.

Semelhante para par.

Análise da L-redução apresentada na aula.

Teorema: Para qq constante $\alpha > 13/14 \approx 0{,}928$, se existe α-aproximação para MAX E3SAT, então P $=$ NP.

L-redução para MAX E3SAT

Cada restrição ímpar(x_1, x_2, x_3) corresponde às cláusulas
$x_1 \vee x_2 \vee x_3$, $\bar{x}_1 \vee \bar{x}_2 \vee x_3$, $\bar{x}_1 \vee x_2 \vee \bar{x}_3$, $x_1 \vee \bar{x}_2 \vee \bar{x}_3$.

Se ímpar$(x_1, x_2, x_3) = 1$,
4 cláusulas satisfeitas, senão apenas 3.

Semelhante para par.

Análise da L-redução apresentada na aula.

Teorema: Para qq constante $\alpha > 13/14 \approx 0{,}928$, se existe α-aproximação para MAX E3SAT, então P $=$ NP.

Teorema: Para qq constante $\alpha > 7/8 \approx 0{,}875$, se existe α-aproximação para MAX E3SAT, então P $=$ NP.

Unique Games

Tipo particular de *constraint satisfaction problem*:

- as variáveis podem não ser binárias;
- as restrições são sempre sobre duas variáveis;

Unique Games

Tipo particular de *constraint satisfaction problem*:

- as variáveis podem não ser binárias;
- as restrições são sempre sobre duas variáveis;
- para cada restrição $f(x_i, x_j)$, cada valor de x_i (ou x_j) determina um único valor de x_j (ou x_i) tq $f(x_i, x_j) = 1$.

Unique Games

Tipo particular de *constraint satisfaction problem*:

- as variáveis podem não ser binárias;
- as restrições são sempre sobre duas variáveis;
- para cada restrição $f(x_i, x_j)$, cada valor de x_i (ou x_j) determina um único valor de x_j (ou x_i) tq $f(x_i, x_j) = 1$.

Variáveis assumem valores em um conjunto $L = [k]$.

Unique Games

Tipo particular de *constraint satisfaction problem*:

- as variáveis podem não ser binárias;
- as restrições são sempre sobre duas variáveis;
- para cada restrição $f(x_i, x_j)$, cada valor de x_i (ou x_j) determina um único valor de x_j (ou x_i) tq $f(x_i, x_j) = 1$.

Variáveis assumem valores em um conjunto $L = [k]$.

Formulação com um grafo:

- um vértice para cada variável
- aresta uv para cada restrição f sobre variáveis u e v
- permutação $\pi_{uv} : L \to L$ para cada aresta uv tq $\pi_{uv}(i) = j$ se $f(i, j) = 1$.

Unique Games

Formulação com um grafo:

- um vértice para cada variável
- aresta uv para cada restrição f sobre variáveis u e v
- permutação $\pi_{uv} : L \to L$ para cada aresta uv tq $\pi_{uv}(i) = j$ se $f(i, j) = 1$.

Unique Games

Formulação com um grafo:

- um vértice para cada variável
- aresta uv para cada restrição f sobre variáveis u e v
- permutação $\pi_{uv} : L \to L$ para cada aresta uv tq $\pi_{uv}(i) = j$ se $f(i, j) = 1$.

Objetivo: encontrar atribuição de valores de L aos vértices que satisfaça o maior número de arestas possíveis.

Unique Games

Formulação com um grafo:

- um vértice para cada variável
- aresta uv para cada restrição f sobre variáveis u e v
- permutação $\pi_{uv} : L \to L$ para cada aresta uv tq $\pi_{uv}(i) = j$ se $f(i, j) = 1$.

Objetivo: encontrar atribuição de valores de L aos vértices que satisfaça o maior número de arestas possíveis.

Aresta uv é satisfeita se $f(i, j) = 1$ onde i é o rótulo de u e j é o rótulo de v.

Unique Games

Formulação com um grafo:

- um vértice para cada variável
- aresta uv para cada restrição f sobre variáveis u e v
- permutação $\pi_{uv} : L \to L$ para cada aresta uv tq $\pi_{uv}(i) = j$ se $f(i, j) = 1$.

Objetivo: encontrar atribuição de valores de L aos vértices que satisfaça o maior número de arestas possíveis.

Aresta uv é satisfeita se $f(i, j) = 1$ onde i é o rótulo de u e j é o rótulo de v.

Decidir se todas as arestas podem ser satisfeitas é fácil!

Unique Games

Formulação com um grafo:

- um vértice para cada variável
- aresta uv para cada restrição f sobre variáveis u e v
- permutação $\pi_{uv} : L \to L$ para cada aresta uv tq $\pi_{uv}(i) = j$ se $f(i,j) = 1$.

Objetivo: encontrar atribuição de valores de L aos vértices que satisfaça o maior número de arestas possíveis.

Unique Games Conjecture (UGC): Para todo ϵ, $\delta > 0$, existe constante $k = k(\epsilon, \delta) > 0$ tq é NP-difícil, dada uma instância I com $|L| = k$ e m arestas, decidir se $\mathrm{OPT}(I) \geq (1-\epsilon)m$ ou $\mathrm{OPT}(I) \leq \delta\, m$.

MAX 2LIN(k) e LUGC

$L = \{0, \ldots, k-1\}$

Para cada aresta uv, temos $c_{uv} \in L$ tq $\pi_{uv}(i) = i - c_{uv} \bmod k$.
(uv satisfeita sse recebe rótulos ij tq $i - j = c_{uv} \bmod k$)

MAX 2LIN(k) e LUGC

$L = \{0, \ldots, k-1\}$

Para cada aresta uv, temos $c_{uv} \in L$ tq $\pi_{uv}(i) = i - c_{uv} \bmod k$.
(uv satisfeita sse recebe rótulos ij tq $i - j = c_{uv} \bmod k$)

Linear Unique Games Conjecture (LUGC): Para todo ϵ, $\delta > 0$, existe constante $k = k(\epsilon, \delta) > 0$ tq é NP-difícil, dada uma instância I do MAX 2LIN(k) com m arestas, decidir se $\mathrm{OPT}(I) \geq (1-\epsilon)m$ ou $\mathrm{OPT}(I) \leq \delta\, m$.

MAX 2LIN(k) e LUGC

$L = \{0, \ldots, k-1\}$

Para cada aresta uv, temos $c_{uv} \in L$ tq $\pi_{uv}(i) = i - c_{uv} \bmod k$.
(uv satisfeita sse recebe rótulos ij tq $i - j = c_{uv} \bmod k$)

Linear Unique Games Conjecture (LUGC): Para todo ϵ, $\delta > 0$, existe constante $k = k(\epsilon, \delta) > 0$ tq é NP-difícil, dada uma instância I do MAX 2LIN(k) com m arestas, decidir se $\mathrm{OPT}(I) \geq (1-\epsilon)m$ ou $\mathrm{OPT}(I) \leq \delta\, m$.

Relação entre o MAX 2LIN(k) e o Problema do Multicorte

Instância do MAX 2LIN(k): $(G = (V, E), k, c)$

MAX 2LIN(k) e LUGC

$L = \{0, \ldots, k-1\}$

Para cada aresta uv, temos $c_{uv} \in L$ tq $\pi_{uv}(i) = i - c_{uv} \bmod k$.
(uv satisfeita sse recebe rótulos ij tq $i - j = c_{uv} \bmod k$)

Linear Unique Games Conjecture (LUGC): Para todo ϵ, $\delta > 0$, existe constante $k = k(\epsilon, \delta) > 0$ tq é NP-difícil, dada uma instância I do MAX 2LIN(k) com m arestas, decidir se $\mathrm{OPT}(I) \geq (1-\epsilon)m$ ou $\mathrm{OPT}(I) \leq \delta\, m$.

Relação entre o MAX 2LIN(k) e o Problema do Multicorte

Instância do MAX 2LIN(k): $(G = (V, E), k, c)$

Instância do multicorte: $G' = (V', E')$ com $V' = V \times L$ e aresta em E' entre (u, i) e (v, j) sse $uv \in E$ e $i - j = c_{uv}$
Pares $s = (u, i)$ e $t = (u, j)$ para todo $u \in V$ e $i \neq j$.

MAX 2LIN(k) e Multicorte

Instância do MAX 2LIN(k): $I = (G = (V, E), k, c)$

Instância I' do multicorte: $G' = (V', E')$ com $V' = V \times L$ e aresta em E' entre (u, i) e (v, j) sse $uv \in E$ e $i - j = c_{uv}$
Pares $s = (u, i)$ e $t = (u, j)$ para todo $u \in V$ e $i \neq j$.

MAX 2LIN(k) e Multicorte

Instância do MAX 2LIN(k): $I = (G = (V, E), k, c)$

Instância I' do multicorte: $G' = (V', E')$ com $V' = V \times L$ e aresta em E' entre (u, i) e (v, j) sse $uv \in E$ e $i - j = c_{uv}$
Pares $s = (u, i)$ e $t = (u, j)$ para todo $u \in V$ e $i \neq j$.

Lema 1: Para cada ϵ, $0 \leq \epsilon \leq 1$, dada qq solução viável de I que satisfaça pelo menos $(1 - \epsilon)|E|$ arestas, existe solução viável de I' de custo no máximo $\epsilon|E'|$.

(Prova vista em aula.)

MAX 2LIN(k) e Multicorte

Instância do MAX 2LIN(k): $I = (G = (V, E), k, c)$

Instância I' do multicorte: $G' = (V', E')$ com $V' = V \times L$ e aresta em E' entre (u, i) e (v, j) sse $uv \in E$ e $i - j = c_{uv}$
Pares $s = (u, i)$ e $t = (u, j)$ para todo $u \in V$ e $i \neq j$.

Lema 1: Para cada ϵ, $0 \leq \epsilon \leq 1$, dada qq solução viável de I que satisfaça pelo menos $(1 - \epsilon)|E|$ arestas, existe solução viável de I' de custo no máximo $\epsilon|E'|$.

(Prova vista em aula.)

Lema 2: Para cada ϵ, $0 \leq \epsilon \leq 1$, dada qq solução viável de I' de custo no máximo $\epsilon|E'|$, existe solução viável de I que satisfaz pelo menos $(1 - 2\epsilon)|E|$ arestas.

(Um pouco mais complicado... Só comentários.)

Consequência

Instância do MAX 2LIN(k): $I = (G = (V, E), k, c)$

Instância I' do multicorte: $G' = (V', E')$ com $V' = V \times L$ e aresta em E' entre (u, i) e (v, j) sse $uv \in E$ e $i - j = c_{uv}$
Pares $s = (u, i)$ e $t = (u, j)$ para todo $u \in V$ e $i \neq j$.

Corolário: Dada UGC, para qq constante $\alpha \geq 1$, não existe α-aproximação para o Multicorte a menos que P $=$ NP.

Consequência

Instância do MAX 2LIN(k): $I = (G = (V, E), k, c)$

Instância I' do multicorte: $G' = (V', E')$ com $V' = V \times L$ e aresta em E' entre (u, i) e (v, j) sse $uv \in E$ e $i - j = c_{uv}$
Pares $s = (u, i)$ e $t = (u, j)$ para todo $u \in V$ e $i \neq j$.

Corolário: Dada UGC, para qq constante $\alpha \geq 1$, não existe α-aproximação para o Multicorte a menos que P $=$ NP.

Teorema: Dada UGC, não existe α-aproximação para o MAX CUT com constante α tq

$$\alpha > \min_{-1 \leq x \leq 1} \frac{\frac{1}{\pi} \arccos(x)}{\frac{1}{2}(1 - x)} \geq 0{,}878,$$

a menos que P $=$ NP.

MAX CUT

Versão equivalente da UGC:

Bipartite Unique Games Conjecture (BUGC): UGC para G bipartido com m arestas e todo vértice de um lado da bipartição com mesmo grau.

MAX CUT

Versão equivalente da UGC:

Bipartite Unique Games Conjecture (BUGC): UGC para G bipartido com m arestas e todo vértice de um lado da bipartição com mesmo grau.

Teorema: Dada BUGC, para qq cte $\gamma > 0$ e $\rho \in (-1, 0)$, NP $\subseteq \mathrm{PCP}(\log n, 2)$, onde completude é $\geq \frac{1}{2}(1-\rho) - \gamma$ e robutez é $\leq \frac{1}{\pi}\arccos(\rho) + \gamma$, e aceita se os 2 bits diferem.

MAX CUT

Versão equivalente da UGC:

Bipartite Unique Games Conjecture (BUGC): UGC para G bipartido com m arestas e todo vértice de um lado da bipartição com mesmo grau.

Teorema: Dada BUGC, para qq cte $\gamma > 0$ e $\rho \in (-1, 0)$, NP $\subseteq \mathrm{PCP}(\log n, 2)$, onde completude é $\geq \frac{1}{2}(1-\rho) - \gamma$ e robutez é $\leq \frac{1}{\pi}\arccos(\rho) + \gamma$, e aceita se os 2 bits diferem.

Teorema: Dada UGC, a menos que P $=$ NP, não existe α-aproximação para o MAX CUT com constante α tq

$$\alpha > \min_{-1 \leq x \leq 1} \frac{\frac{1}{\pi}\arccos(x)}{\frac{1}{2}(1-x)} \geq 0{,}878.$$

Fim...

OBRIGADA PELA ATENÇÃO!!!