ALGORITMOS DE APROXIMACIÓN PARA PROBLEMAS NP DUROS

MARITZA HERRERA FLOREZ
YUDY MARCELA BOLAÑOS RIVERA

UNIVERSIDAD DEL CAUCA
FACULTAD DE CIENCIAS NATURALES, EXACTAS Y DE LA EDUCACIÓN
DEPARTAMENTO DE MATEMÁTICAS
2006

LAS CLASES NPO y PO

Un problema de optimización es una cuádrupla $A = \langle I, sol, cost, obj \rangle$ donde:

1. $I \leftarrow$ conjunto de instancias de A.
2. $sol(x) \leftarrow$ conjunto de soluciones factibles de una instancia x de I.
3. $cost$ es una función tal que:
   $cost(y) \in \mathbb{Z}^{+} \leftarrow$ costo o valor de la solución $y \in sol(x)$ para cualquier instancia x en I.
4. $obj \in \{\text{máx}, \text{mín}\}$.

LAS CLASES NPO y PO

La clase PO es el conjunto de todos los problemas de optimización en NPO que tienen un algoritmo de tiempo polinomial que lo re-suelve.

EL PROBLEMA DEL AGENTE VIAJERO (MIN-AV)

INSTANCIA:	Un conjunto $C = \{c_1,\, c_2, \ldots, c_m\}$ de ciudades y una distancia $d(c_i,\, c_j) \in \mathbb{Z}^+$ para cada par de ciudades $c_i,\, c_j \in C$.
SOLUCION:	Un $tour$ de todas las ciudades en C, esto es, un ordenamiento $\langle c_{\pi(1)},\, c_{\pi(2)}, \ldots, c_{\pi(m)} \rangle$ donde π es una permutación $\pi : \{1, \ldots, m\} \to \{1, \ldots, m\}$.
COSTO:	La longitud del $tour$, esto es, $d(c_{\pi(m)},\, c_{\pi(1)}) + \sum_{i=1}^{m-1} d(c_{\pi(i)},\, c_{\pi(i+1)})$.
OBJETIVO:	Minimizar.

EL PROBLEMA DEL AGENTE VIAJERO MÉTRICO (MIN-AVM)

INSTANCIA:	Un conjunto $C = \{c_1, c_2, \ldots, c_m\}$ de ciudades y una distancia $d(c_i, c_j) \in \mathbb{Z}^+$ para cada par de ciudades $c_i, c_j \in C$ que satisface la desigualdad triangular.
SOLUCIÓN:	Un $tour$ de C.
COSTO:	La longitud del $tour$.
OBJETIVO:	Minimizar.

Resolver un problema de optimización $A = \langle I, sol, cost, obj \rangle$ consiste en encontrar para cada $x \in I$, una solución $y \in sol(x)$ tal que

$$\begin{aligned} cost(y) &= obj\{cost(z) : z \in sol(x)\} \\ &= OPT \end{aligned}$$

Problemas de decisión subyacentes

Si $A = \langle I,\ sol,\ cost,\ obj\rangle$ es un problema de optimización en NPO, entonces su problema de decisión subyacente viene dado por:

INSTANCIA: $x \in I$, $k \in \mathbb{Z}$.
PREGUNTA: ¿Existe $y \in sol(x)$, tal que $cost(y) \leq k$? (si obj = mín)
¿Existe $y \in sol(x)$, tal que $cost(y) \geq k$? (si obj = máx)

Algunos resultados

- Si un problema de optimización pertenece a NPO, entonces el problema de decisión subyacente pertenece a NP.
- Si un problema de optimización pertenece a PO entonces el problema de decisión subyacente pertenece a P.
- Si $P \neq NP$ entonces cualquier problema de optimización en NPO cuyo problema subyacente es NP-completo no pertenece a PO.
- Si P$\neq NP$ entonces $PO \neq NPO$.

CLASES DE APROXIMACIÓN

Factor de aproximación

Sea $A = \langle I,\ sol,\ cost,\ obj \rangle$ un problema en NPO. Dada una instancia $x \in I$ y una posible solución $y \in sol(x)$, se define el *factor de aproximación* de y con respecto a x como:

$$R(x, y) = \text{máx}\left\{\frac{cost(y)}{OPT}, \frac{OPT}{cost(y)}\right\}$$

Algoritmo *r(n)*-aproximado

Sea $A = \langle I, sol, cost, obj \rangle$ un problema en NPO y sea T un algoritmo tal que, para toda instancia $x \in I$, retorna una solución factible $T(x) \in sol(x)$. Dada una función arbitraria $r : \mathbb{N} \to \mathbb{Q}^+$, se dice que T es un algoritmo $r(n)$-aproximado para A, si para cualquier instancia x, el factor de aproximación de la solución $T(x)$ con respecto a x verifica la siguiente desigualdad:

$$R(x, T(x)) \leq r(|x|).$$

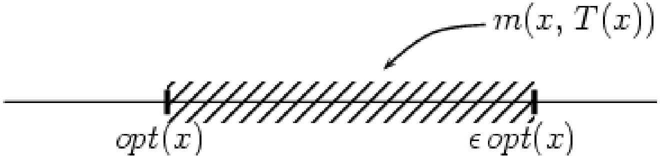
Clase APX

Sea A un problema en NPO. A pertenece a la clase APX si es ϵ-aproximable para alguna constante $\epsilon > 1$.

Si A es un problema de minimización entonces

$$R(x, T(x)) = \frac{cost(T(x))}{OPT} \leq \epsilon$$

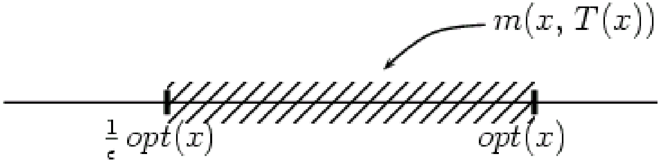
y por lo tanto, $cost(T(x)) \leq \epsilon\, OPT$.



Si A es un problema de maximización entonces

$$R(x,\, T(x)) = \frac{OPT}{cost(T(x))} \leq \epsilon$$

y por lo tanto, $\frac{1}{\epsilon} OPT \leq cost(T(x))$, con $\epsilon > 1$.



Esquema de aproximación

Sea A un problema en NPO. Un algoritmo T es un esquema de aproximación para A si, para toda instancia x de A y para cualquier racional $\epsilon > 1$, T(x, ϵ) retorna una solución posible de x cuyo factor de aproximación es a lo más ϵ.

Clase PTAS

Un problema A en NPO pertenece a la clase PTAS si admite un esquema de aproximación tiempo polinomial, esto es, un esquema de aproximación cuyo tiempo de complejidad es acotado por $q(|x|)$ donde q es un polinomio en el tamaño de x.

Ejemplo: $2^{1/(\epsilon-1)}\, p\left(|x|\right)$ donde p es un polinomio,

$|x|^{1/(\epsilon-1)}$.

Clase FPTAS

Un problema A en NPO pertenece a la clase FPTAS si admite un esquema de aproximación totalmente polinomial, esto es, un esquema de aproximación cuyo tiempo de complejidad es acotado por $q\left(|x|, \frac{1}{(\epsilon-1)}\right)$ donde q es un polinomio tanto en $|x|$ como en $\frac{1}{(\epsilon-1)}$.

Ejemplo: $\frac{1}{\epsilon-1}\ p\left(|x|\right)$ donde p es un polinomio,

$\left(\frac{1}{\epsilon-1}\right)^2 |x|^3$.

CICLO HAMILTONIANO (HC)

INSTANCIA : Un grafo G=(V, E)

PREGUNTA : ¿ Existe un ordenamiento $\langle v_1, v_2, \ldots, v_n \rangle$ de V, con $n = |V|$ tal que $\{v_i, v_{i+1}\} \in E$ para $1 \leq i < n$ y $\{v_n, v_1\} \in E$?. $\diamondsuit$

Teorema 1. Si P $\neq$ NP entonces existe un problema en NPO que no es aproximable.

Demostración

- Se probará que:
  Si P $\neq$ NP y $\epsilon > 1$, no existe un algoritmo ϵ-aproximado para el problema del Agente Viajero (MIN-AV).

- Sea $G = (V, E)$ una instancia de HC y sea A un algoritmo ϵ-aproximado tiempo polinomial para MIN-AV, para algún $\epsilon > 1$ y entero.

- Se transforma G en una instancia del AV:

  Sea $G' = (V, E')$ el grafo completo con conjunto de vértices V y $E' = \{\{u, v\} : u, v \in V \text{ y } u \neq v\}$.

- Se asigna un costo entero a cada arista en E':

$$c(u,\, v) = \begin{cases} 1 & \text{si } \{u,\, v\} \in E \\ \epsilon\,|V| + 1 & \text{en otro caso} \end{cases}$$

- Se considera la instancia del problema del Agente Viajero $(G',\, c)$ es una instancia de AV.

  Si G tiene un ciclo hamiltoniano H, $(G',\, c)$ contiene un *tour* de costo $|V|$.

  Si G no contiene un ciclo hamiltoniano, entonces cualquier *tour* de G' debe usar alguna arista que no está en E. Pero cualquier *tour* que use alguna arista que no está en E tiene un costo de al menos $(\epsilon\,|V| + 1) + (|V| - 1) > \epsilon|V|$.

- Se aplica el algoritmo de aproximación A a la instancia (G', c) de MIN-AV.

  Si G contiene un ciclo hamiltoniano, A debe retornar un $tour$ de costo $|V|$ que es el ciclo hamiltoniano en G.

  Si G no contiene un ciclo hamiltoniano, A retorna un *tour* de costo mayor que $\epsilon\,|V|$.

Ciclo Euler $\leftarrow$ Camino a través de un grafo el cual comienza y finaliza en el mismo vértice e incluye cada arista exactamente una sola vez.

Grafo Euleriano $\leftarrow$ Grafo que tiene un ciclo Euler.

Matching $\leftarrow$ Dado un grafo $G = (U, F)$, un subconjunto de las aristas $M \subseteq F$ es un $matching$ si ningún par de aristas de M comparten un punto final.

Matching perfecto $\leftarrow$ Matching que cubre todos los vértices de un grafo.

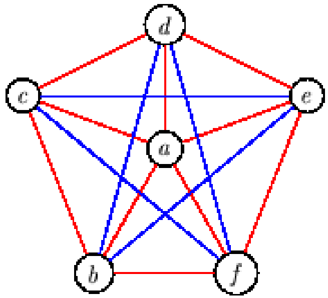
Teorema 2. El problema del AGENTE VIAJERO MÉTRICO (MIN-AVM) pertenece a APX.

Demostración

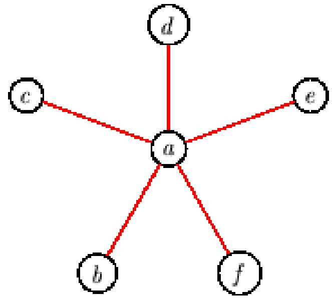
Para desarrollar un algoritmo de aproximación para el Agente Viajero Métrico se modelará el problema como un grafo completo no dirigido $G = (V, E)$, donde el conjunto V de vértices representa el conjunto de ciudades y un costo entero no negativo asociado a cada arista $\{u, v\} \in E$ que representa la distancia entre cada par de ciudades.

Algoritmo APROX-AVM (G, c)-factor 2

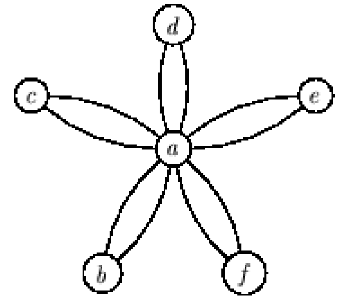
1. Encuentra un árbol de recubrimiento mínimo T para G
2. Dobla cada arista de T para obtener un grafo Euleriano.
3. Encuentra un $tour$ Euler W sobre el grafo del paso 2.
4. Retorna el $tour$ que visita los vértices de G en el orden de su primera aparición en W. Sea H este $tour$.



Grafo G=(V,E)



1. Árbol de rec. min. para G

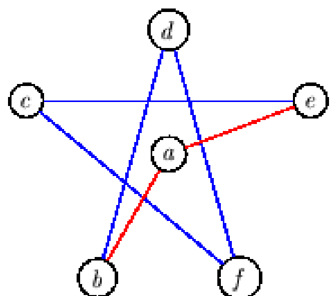


2. Grafo Euleriano

3. Tour Euler

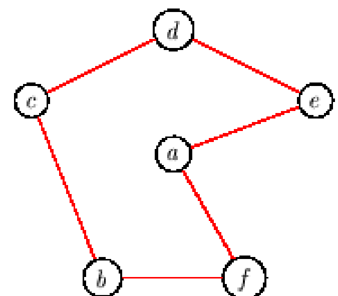
W=$\langle a, e, a, c, a, f, a, d, a, b, a\rangle$

H=$\langle a, e, c, f, d, b\rangle$



4. Tour retornado por APROX AVM-fact-2

Cost(H)=10



Tour óptimo

OPT=6

Sea H^* un *tour* óptimo para el grafo $G = (V,\ E)$ dado.

Se probará $cost(H) \leq 2\,cost(H^*)$, donde H es el *tour* retornado por APROX-AVM.

1. Si T es un árbol de recubrimiento mínimo para V, entonces

$$cost(T) \leq cost(H^*) \tag{1}$$

2. Dobla cada arista de T obteniendo de este modo un grafo en el que todo vértice tiene grado par. Además un grafo tiene un $tour$ Euler si y sólo si todos sus vértices tienen grado par; por tanto, el grafo obtenido es un grafo Euleriano.

3.Sea W un $tour$ Euler de este grafo. Puesto que W visita cada arista de T exactamente dos veces, entonces

$$cost(W) = 2\,cost(T) \tag{2}$$

De (1) y (2) se tiene

$$cost(W) \leq 2\,cost(H^*) \tag{3}$$

Desafortunadamente, W no es generalmente una solución para MIN-AVM ya que un $tour$ Euler puede visitar algunos vértices más de una vez. Sin embargo, por la desigualdad triangular se puede borrar una visita a cualquier vértice y la longitud no se incrementa. Repitiendo este proceso, se pueden remover de W todas, menos las primeras visitas a cada vértice.

4.De esta manera se obtiene el $tour$ H retornado por APROX-AVM-factor2 tal que:

$$cost(H) \leq cost(W) \tag{4}$$

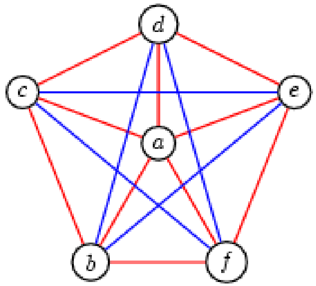
Finalmente de (3) y (4) se obtiene que

$$cost(H) \leq 2\,cost(H^*)$$

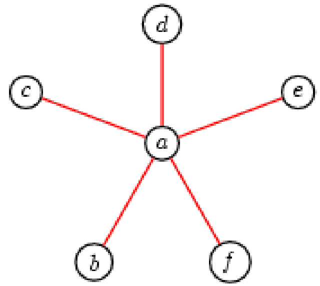
MEJORANDO EL FACTOR A 3/2

Algoritmo APROX-AVM (G, c)-factor 3/2

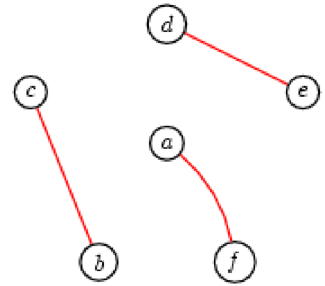
1. Encuentra un árbol de recubrimiento mínimo T para G, usando PRIM(G, c, r).
2. Computa un Matching Perfecto de costo mínimo M, en el conjunto de vértices de grado impar de T
3. Adiciona M a T y obtiene un grafo Euleriano.
4. Encuentra un tour Euler W, de este grafo.
5. Devuelve el tour H, que visita los vértices de G en el orden de su primera aparición en W.



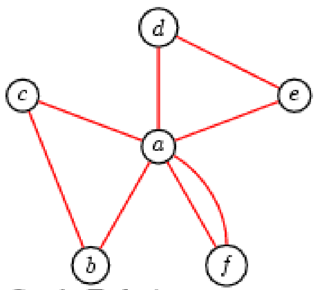
Grafo G=(V,E)



1.Árbol de rec. min. para G



2. Matching Perfecto para V'

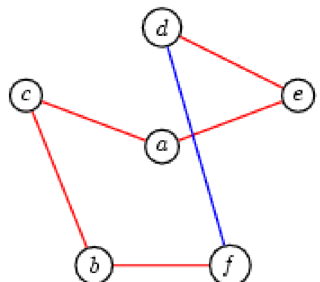


3. Grafo Euleriano

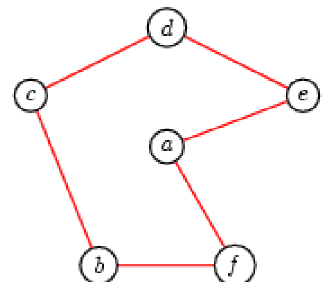
4. Tour Euler

W=$\langle a, e, d, a, f, a, b, c, a \rangle$

H=$\langle a, e, d, f, b, c \rangle$



5. Tour retornado por
APROX AVM-fact-3/2
Cost(H)=7



Tour óptimo
OPT=6

Lema 1. Sea $V' \subseteq V$ tal que $\mid V' \mid$ es par, y sea M un $matching$ perfecto de costo mínimo sobre V'. Entonces,

$$cost(M) \leq \frac{OPT}{2}$$

Demostración

Sea τ un $tour$ óptimo de G.

Sea τ' el $tour$ sobre V' obtenido acortando τ.

Por la desigualdad triangular,

$$cost(\tau') \leq cost(\tau)$$

$\mid \tau' \mid$ es par puesto que $\mid V' \mid$ es par y $\mid V' \mid = \mid \tau' \mid$.
Además, todo grafo con un número par de vértices posee un $matching$ perfecto.

τ' es la unión de dos $matchings$ perfectos sobre V', cada uno consistente de aristas alternadas de τ'.

Sea M el $matching$ perfecto de costo mínimo entre los dos.

$$cost(M) \leq \frac{cost(\tau')}{2}$$
$$\frac{cost(\tau)}{2} = OPT$$

Teorema 3. El algoritmo APROX-AVM factor 3/2 es un algoritmo 3/2-aproximado para el problema del agente viajero métrico (MIN-AVM).

Demostración

1. Sea T el árbol de recubrimiento mínimo para el conjunto de vértices del grafo G.

2. Sea V' el conjunto de vértices de grado impar de T.

La suma de los grados de todos los vértices en el árbol de recubrimiento mínimo T es par, esto se debe a que cada arista incide exactamente sobre dos vértices y en consecuencia en la suma de los grados cada arista se cuenta dos veces.

$\mid V' \mid$ es par puesto que de lo contrario al sumar los grados de todos los vértices de T, se suma una cantidad par (sumgrado(V-V')) y una cantidad impar (sumgrado(V')) lo cual implicaría que la suma de los grados de los vértices de T seria impar.

Luego, V' tiene un $matching$ perfecto. Sea M el $matching$ perfecto de costo mínimo.

3. Adiciona M a T y se obtiene un grafo con todos los vértices de grado par puesto que se ha adicionado una arista entre vértices de grado impar, las aristas de M. Como $\mid M \mid = \mid V' \mid$, esto garantiza que todos los vértices en V' poseen ahora grado par. De este modo se obtiene un grafo Euleriano.

4. Sea W un $tour$ Euler de este grafo.

$$\begin{aligned} cost(W) &\leq cost(T) + cost(M) \\ &\leq OPT + \tfrac{1}{2}OPT \\ &= \tfrac{3}{2}OPT \end{aligned} \tag{5}$$

5. Eliminando de W los vértices repetidos se obtiene el $tour$ H que retorna el algoritmo.

Por la desigualdad triangular y por lo anterior, se tiene que:

$$cost(H) \leq cost(W) \tag{6}$$

Por lo tanto, de 5 y 6 resulta:

$$cost(H) \leq \frac{3}{2}OPT$$

Definición 1. (Verificador (log n, 1)-restringido)

Un verificador $(\log n, 1)-$ restringido V es una máquina de Turing (probabilística) tiempo polinomial que tiene acceso a una entrada x, una cadena r compuesta de $O(\log|x|)$ ***bits random*** y una $prueba$ Π. Dependiendo de la entrada x, la cadena *random* r y la $prueba$ Π, el verificador V aceptará o rechazará la entrada x. Aunque el verificador tiene acceso completo a la entrada x, el acceso a la $prueba$ Π es muy limitado, V solamente examina un número constante de $bits$ de Π.

Teorema 4 (Teorema PCP). Un lenguaje L está en NP si y sólo si L es decidible por un verificador $(\log n, 1)$-restringido.

Teorema 5. Si P $\neq$ NP entonces existe un problema en APX que no pertenece a PTAS.

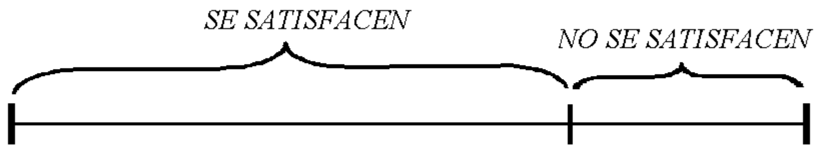
***Demostración*.**

- Se probará que, si P $\neq$ NP entonces MAX3SAT $\in$ APX no admite un PTAS.

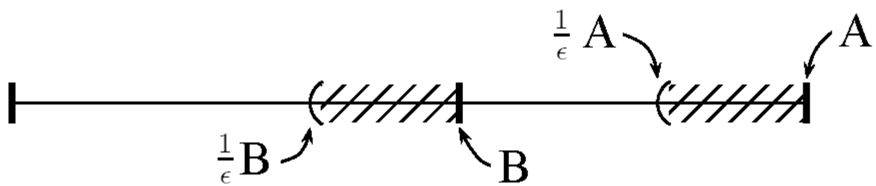
EL PROBLEMA MAX3SAT

- INSTANCIA: Una colección $C = \{c_1, c_2, \ldots, c_m\}$ de cláusulas sobre un conjunto finito U de variables tales que $|c_i| = 3$ para $1 \leq i \leq m$.
- SOLUCIÓN: Una asignación de verdad para U.
- MEDIDA: Número de cláusulas que se satisfacen con la asignación de verdad.
- OBJETIVO: Maximizar.

- Se toma un lenguaje arbitrario L de NP.
- Se aplica el verificador V sobre una entrada del lenguaje L para construir una instancia de 3SAT.



- Se ejecuta el PTAS A_ϵ sobre la instancia de 3SAT



□

Teorema 6.

$$FPTAS \subseteq PTAS \subseteq APX \subseteq NPO$$

y las inclusiones son estrictas si y sólo si P $\neq$ NP.