Nociones de **aproximabilidad** (2w = 4t = 360mns)

Jérémy Barbay

5 de junio de 2012

- problemas que son o no aproximables
- Referencia: Cap 35, "Approximation Algorithms", pag 1022, CLRS 2nd Edition

1. (Motivacion)

Aproximacion de problemas de optimizacion NP-dificiles

- Que problemas NP dificiles conocen?
  - colorisacion de grafos
  - ciclo hamiltonian
  - Recubrimiento de Vertices (Vertex Cover)
  - Bin Packing
  - Problema de la Mochila
  - Vendedor viajero (Traveling Salesman)
- Que hacer cuando se necessita una solucion en tiempo polinomial?
  - Consideramos los problemas NP completos de decision, generalmente de optimizacion. Si se necesita una soluciono en tiempo polinomial, se puede considerar una aproximacion.

2. $p(n)$-aproximacion

- Dado un problema de minimizacion, un algoritmo A es un $p(n)$**-aproximacion** si

$$\forall n \max_x \frac{C_A(x)}{C_{OPT}(x)} \leq p(n)$$

- Dado un problema de maximizacion, un algoritmo A es un $p(n)$**-aproximacion** si

$$\forall n \max_x \frac{C_{OPT}(x)}{C_A(x)} \leq p(n)$$

- Notas:
  1. Aqui consideramos la cualidad de la solucion, NO la complejidad del algoritmo. Usualmente el problema es NP-dificil, y el algoritmo de aproximacion es de complejidad polinomial.
  2. Las definiciones estan escritas de manera que las fracciones sean mayor que 1.
  3. A veces consideramos tambien $C_A(x) - C_{OPT}(x)$ (minimizacion) y $C_{OPT}(x) - C_A(x)$.

- Si $C_A(x) - C_{OPT}(x) < \varepsilon \forall x$,
- que se puede decir de $\frac{C_A(x)}{C_{OPT}(x)}$?
  - $\frac{C_A(x)}{C_{OPT}(x)}$
    - $< \frac{C_{OPT}(x)+\varepsilon}{C_{OPT}(x)}$
    - $= 1 + \frac{\varepsilon}{C_{OPT}(x)}$
    - $\to 1$ cuando $C_{OPT}(x)$ crece al infinito.

2.1. Ejemplo: Bin Packing (un problema que es 2-aproximable)

- Referencia:
  - CLRS2, pagina 1049
- DEFINICION

  Dado n valores $x_1, \ldots, x_n$, $0 \leq x_i \leq 1/2$, cual es la menor cantidad de cajas de tamaño 1 necesarias para empaquetarlas?
- Algoritmo `Greedy`
  - Considerando los x_i en orden,
  - llenar la caja actual todo lo posible,
  - pasar a la siguiente caja.
- Analisis
  - Este algoritmo tiene complejidad lineal $O(n)$.
  - `Greedy` da una 2-aproximacion.
  - (se puede mostrar facilamente en las instancias donde el algoritmo optima llene completamente todas las cajas.)
    - (interaccion)
- Ademas, hay mejores aproximaciones,
  1. `Best-fit` tiene performance ratio de 1.7 en el peor caso
  2. Extract from "Best-Fit Bin-Packing with Random Order (1997), Kenyon"

     Best-fit is the best known algorithm for on-line binpacking, in the sense that no algorithm is known to behave better both in the worst case (when Best-fit has performance ratio 1.7)
  3. Un otro paper donde particionan los x_i en tres classes, placeando los x_i mas grande primero, buscando el placamiento optimal de los x_i promedio, y usando un algoritmo greedy para los x_i pequenos. [Karpinski?]
  4. la distribucion de los tamanos de las valores hacen la instancia dificil o facil.

2.2. Ejemplo: "Vertex Cover" (Recubrimiento de Vertices)

- REFERENCIA: CLRS2, pagina 1024
- DEFINICION:
  - Dado un grafo $G = (V, E)$,
  - V' es un *vertex cover* de G si
    - $\forall e = (u, v) \in E$, $u \in V'$ o $v \in V'$)
  - V' es un *vertex cover optima* de G si

    - V' es de tamano minima.
  - Cual es el menor $V' \subseteq V$ tal que
  - V' sea un *vertex cover optima* de G?
- Algoritmo de aproximación:
  - $V' \leftarrow \emptyset$
  - while $E \neq \emptyset$
    - sea $(u, v) \in E$
    - $V' \leftarrow V' \cup u, v$
    - $E \leftarrow E\ (x, y) talquex = uox = voy = uoy = v$
  - return V'
- Discusion:
  - Cual es la complejidad del algoritmo?
  - es una 2-aproximacion o no?
- LEMA: El algoritmo es una 2-aproximacion.
  - PRUEBA:
    - Cada par u, v que la aproximacion elige esta conectada, entonces u o v estas en cualquier soluciono optima de Vertex Cover.
    - Como se eliminan las aristas incidentes en u y v, los siguientes pares que se eligen no tienen interseccion con el actual, entonces cada 2 nodos que el algoritmo elige, uno pertenece a la solucion optima.
    - quod erat demonstrandum, (QED).

2.3. Ejemplo: "Vertex Cover" con pesos

- DEFINICION
  - Dado
    - $G = (V, E)$ y
    - $c : V^+$,
  - se quiere un
    - $V^* \subseteq V$ que cubra E y
    - que minimice $\sum_{v \in V^*} c(v)$.
- Este problema es NP Completo.
  - exercicio: que los alumnos le prueban.
  - Pregunta:
    - una reduccion a un problema NP completa con esquema de approximacion impliqua un esquema de approximacion o no?
- LEMA: Vertex Cover con pesos es 2-aproximable
- PROOF:

1. Sea variables $x(v) \in 0, 1, \forall v \in V$
   - el costo de V^* sera $\sum x(v)c(v)$
   - objetivo: mín $\sum_{v \in V} x(v)c(v)$ donde $0 \leq x(v) \leq 1 \forall v \in V$

- $x(u) + x(v) \geq 1 \forall u, v \in E$

2. $x(v) \in Z$ sere programacion entera, que es NP Completa $x(v) \in R$ sere programacion lineal, que es polinomial el valor $\sum x(v)c(v)$ que produce el programo lineal es inferior a la mejor solucion al problema de VC.
3. Algoritmo
   - resolver el problema de programacion lineal
     - nos da $x(v_1), \ldots, x(v_n)$
   - $V^* \leftarrow \emptyset$
   - for $v \in V$
     - si $x(v) \geq 1/2$
       - $V^* \leftarrow V^* \cup v$
   - return V^*
4. Propiedades
   - V^* es un vertex cover:
     - si $\forall (u, v) \in E, x(u) + x(v) \geq 1$
     - entonces, $x(u) \geq 1/2ox(v)/geq1/2$
     - entonces, $u \in V^*ov \in V^*$
   - V^* es una 2-aproximacion:
     - $c(V^*) = \sum_v y(v)c(v)$ donde $y(v) = 1six(v) \geq 1/2$, y 0 sino
     - entonces $y(v) \leq 2x(v)$
     - $\sum y(v)c(v) \leq \sum 2x(v)c(v) \leq 2OPT$
     - $C(V^*) \leq 2OPT$
5. QED

2.4. Ejemplo: Vendedor viajero con inegalidad triangular (Traveling Salesman)

- REFERENCIA: CLRS2, pagina 1027
- DEFINICION:

  Dado $G = (V, E)$ dirigido y $C : E \to R^+$ una funcion de costos, encontrar un recorrido que toque cada ciudad una vez, y minimice la suma de los costos de las aristas recorridas.
- LEMA: Si c satisface la desigualdad triangular $\forall x, y, zc(x, y) + x(y, z) \geq c(x, z)$, hay una 2-aproximacion.
  - PROOF:
    - Algoritmo de Aproximacion (con desigualdad triangular)
      - construir un *arbol cobertor minimo* ("Minimum Spanning Tree" MST)
      - se puede hacer en tiempo polinomial, con programacion lineal.
      - $C_{MST} \leq C_{OPT}$
      - producimos un recorrido en profundidad "Deep First Search" (DFS) del MST:
      - $C_{DFS} = 2C_{MST} \leq 2C_{OPT}$
      - (factor dos porque el camino de vuelta puede ser al maximo de tamano igual al tamano del camino de ida)
      - eliminamos los nodos repetidos del camino, que no crece el costo
      - $C_A \leq 2C_{OPT}$
    - quod erat demonstradndum, QED.

2.5. Ejemplo: Vendedor viajero sin inegalidad triangular (Traveling Salesman)

- LEMA: Si c no satisface la desigualdad triangular, el problema de vendedor viajero no es aproximables en tiempo polinomial (a menos que P=NP$).
  - PROOF:
    - Supongamos que existe una $p(n)$-aproximacion de tiempo polinomial.
    - Dado un grafo $G = (V, E)$
    - Construimos un grafo $G' = (V, E')$ tal que
      - $E' = V^2$ (grafo completo), y
      - $c(u, v) = 1$ si $(u, v) \in E$ $np(n)$ sino
    - Si no hay un Ciclo Hamiltonian en E,
      - todas las soluciones usan al menos una arista que no es en E
      - entonces todas las soluciones tienen costo mas que $np(n)$
    - Si hay un Ciclo Hamiltonian en E
      - tenemos una aproximacion de una soluciono de costo menos que $(n-1)p(n)$ QED

3. PTAS y FPTAS

3.1. Definiciones

- Esquema de aproximacion poliniomial **PTAS**

  Un **esquema de aproximacion polinomial** para un problema es un algoritmo A que recibe como input
  - una instancia del problema y
  - un parametro $\varepsilon > 0$

  y produce una $(1 + \varepsilon)$ aproximacion. Para todo ε fijo, el tiempo de A debe ser polinomial en n, el tamaño de la instancia.

- Ejemplos de complejidades: Cuales tienen sentidos?
  - □ $O(n^{2/3})$
  - □ $O(\frac{1}{\varepsilon^2}n^2)$
  - □ $O(2^{\varepsilon}n)$
  - □ $O(2^{1/\varepsilon}n)$

- Esquema de aproximacion completamente polinomial **FPTAS**

  Un **esquema de aproximacion completamente polinomial** es un PTAS donde el tiempo del algoritmo es polinomial en n y en $1/\varepsilon$.

3.2. Ejemplo: Problema de la Mochila

1. Definicion
   - Dado
     - n pesos $p_1, \ldots, p_n \geq 0$,
     - n valores $v_1, \ldots, v_n \geq 0$,
     - un peso total maximo P
   - Queremos encontrar $S \subset [1..n[$ tal que
     - $\sum_{i \in S} p_i \leq P$
     - $\sum_{i \in S} v_i$ sea maximal.
2. Solucion Exacta
   - Dado $L = \{y_1, , \ldots y_m\}$,
     - definimos $L + x = \{y_1 + x, \ldots, y_m + x\}$.
   - Algoritmo:
     - $L \leftarrow \{\emptyset\}$
     - for $i \leftarrow 1$ to n
       - $L \leftarrow$ merge $(L, L + x_i)$
       - prune(L, P) (remudando las valores $> P$)
     - return máx(L)
   - Complejidad del Algoritmo
     - Puede considerar 2^n conjuntos en total!

1. Solucion aproximada (inspirada del algoritmo exacto)
   - Operacion "Recorte"
     - Definimos la operacion de **recorte** de una lista L con parametro δ:
       - Dado $y, z \in L$, z **represente** a y si
         - $y/(1 + \delta) \leq z \leq y$
     - vamos a eliminar de L todos los y que sean representados por alguno z no eliminado de L.
     - Recortar(L, δ)
       - Sea $L = \{y_1, \ldots, y_m\}$
       - $L' \leftarrow \{y_1\}$

- $Last \leftarrow y_1$
- for $i \leftarrow 2$ to m
  - si $y_i > Last(1+\delta)$
  - $L \leftarrow L'.\{y_i\}$
  - $Last \leftarrow y_i$
- return L'

- Algoritmo de Aproximacion:
  - $L \leftarrow \{\emptyset\}$
  - for $i \leftarrow 1$ to n
    - $L \leftarrow merge(L, L + x_i)$
    - $prune(L, t)$ (remudando las valores $> t$)
    - $L \leftarrow recortar(L, \epsilon/2n)$
  - return máx(L)
  - How to choose δ?
    - δ must be proportional to ϵ (smaller value of ϵ means tighter approximation of L)
    - δ must be inversely proportional to n:
      - any error in the recorting will be exponentially amplified by the repetition of the process.
    - $\delta = \epsilon/2n$
- Analysis
  - El resultado es una $(1+\epsilon)$-aproximacion:

    a) retorne una solucion valida, tal que
    - $\sum(S') \leq t$ para algun $S' \subset S$

    b) en el paso i, para todo $z \in L_{OPT}$, existe un $y \in L_A$ tal que z representa a y.
    - Luego de los n pasos, el z^* optimo en L_{OPT} tiene un representante $y^* \in L_A$ tal que
    $$z^*/(1+\epsilon/2n)^n \leq y^* \leq z^*$$

    c) Para mostrar que el algoritmo es una $(1+\epsilon)$ aproximacion,
    - hay que mostrar que
      - $z^*/(1+\epsilon) \leq y^*$
    - entonces, debemos mostrar que
      - $(1+\epsilon/2n)^n \leq 1+\epsilon$

- Eso se muestra con puro calculo:
  - $(1+\epsilon/2n)^n \leq ?1+\epsilon$
  - $e^{n \lg(1+\epsilon/2n)}$
  - $\leq e^{n\epsilon/2n}$ para ϵ sufficiently small
  - $= e^{\epsilon/2}$
  - $\leq ?e^{\ln(1+\epsilon)}$
  - eso es equivalente a elegir ϵ tal que $\epsilon/2 \leq \ln(1+\epsilon)$
  - i.e. cualquier tal que $0 < \epsilon \leq 1$

d) El algoritmo es polinomial (en todos los parametros)

- despues de recortar dedos $y_i, y_{i+1} \in L$ se cumple $y_{i+1} > y_i(1+\delta)$ y el ultimo elemento es $\leq t$
- entonces, la lista contiene $0, 1$ y luego a lo mas $\lfloor \log_{(1+\delta)} t \rfloor$
- entonces el largo de L en cada iteracion no supera $2 + \frac{\log t}{\log(1+\varepsilon/2n)}$
- Nota que $\ln(1+x)$
  - $= -\ln(1/(1+x))$
  - $= -\ln((1+x-x)/(1+x))$
  - $= -\ln(1-x/(1+x))$
  - $= -\ln(1+y) \geq -y$
  - $\geq -(-x/(1+x)) = x/(1+x)$
- Entonces
  - $2 + \frac{\ln t}{\ln 1+\epsilon/2n}$
  - $\leq 2 + ((1+\epsilon/2n)2n \ln t)/\epsilon$
  - $= (2n \ln t)/\epsilon + \ln t + 2$
  - $= O(n \lg t/\epsilon)$
- Entonces cada iteracion toma $O(n \lg t/\epsilon)$ operaciones
- Las n iteraciones en total toman
  - $O(n^2 \lg t/\epsilon)$ operaciones

4. Conclusion/Wrap up Aproximabilidad