Föreläsning 12+13: Approximationsalgoritmer

Många av de **NP**-fullständiga problemen är från början optimeringsproblem: TSP, Graph Coloring, Vertex Cover etc.
Man tror att $\mathbf{P} \neq \mathbf{NP}$ och att det alltså inte går att lösa problemen exakt i polynomisk tid.
En *approximationsalgoritm* till ett optimeringsproblem i **NP** är en algoritm som i polynomisk tid hittar en lösning som garanterat ligger nära den optimala.

Approximation av Vertex Cover

```
ApproxVertexCover(G = (V, E))
(1)   C ← ∅
(2)   while E ≠ ∅
(3)       Ta godtycklig kant (u, v) ∈ E
(4)       C ← C ∪ {u} ∪ {v}
(5)       Ta bort alla kanter i E som innehåller u eller
          v
(6)   return C
```

Algoritmen returnerar alltid en hörntäckning: När en kant togs bort lades minst ett av de ingående hörnen till C.
Betrakta kanten (u, v) i något skede i algoritmen. Minst ett av hörnen u och v måste ingå i alla hörntäckningar, algoritmen låter båda ingå.
$\Rightarrow$ Den resulterande hörntäckningen är högst dubbelt så stor som den optimala.
Tidskomplexitet: $O(|E|)$

Mått på approximerbarhet

Approximationskvoten för en algoritm är
$\max \frac{approx}{opt}$ för minimeringsproblem

$\max \frac{opt}{approx}$ för maximeringsproblem
Den är alltså alltid ≥ 1 med likhet om algoritmen alltid hittar den optimala lösningen.
Annars är den ett mått på hur långt från den optimala lösningen algoritmen kan vara som värst.
Algoritmen för minimal hörntäckning har approximationskvot 2 eftersom den ger en hörntäckning som är högst dubbelt så stor som den minimala.

Grader av approximerbarhet

De **NP**-fullständiga problemen är väsentligen lika svåra att lösa exakt men är olika svåra att approximera:

- För en del problem finns det för varje $\epsilon > 0$ en polynomisk algoritm med approximationskvot $1 + \epsilon$.
  Ex.: Kappsäcksproblemet

- Andra problem kan approximeras inom någon konstant > 1 men inte godtyckligt nära 1 om $\mathbf{P} \neq \mathbf{NP}$.
  Ex.: Minimal hörntäckning
- Det finns problem som inte kan approximeras inom någon konstant om $\mathbf{P} \neq \mathbf{NP}$.
  Ex.: Maximal klick

Approximation av TSP

Visa att TSP$\notin$ APX dvs att TSP inte kan approximeras.
Antag motsatsen dvs att TSP kan approximeras inom faktor B.
Reduktion från Hamiltoncykel:

```
HAMILTONCYKEL(G)
(1)   n ← |V|
(2)   foreach (v_i, v_j) ∈ E
(3)       w(p_i, p_j) ← 1
(4)   foreach (v_i, v_j) ∉ E
(5)       w(p_i, p_j) ← |V|B
(6)   if TSAPPROX(p_i,t) ≤ |V|B
(7)       return TRUE
(8)   return FALSE
```

Om TSAPPROX kan approximera TSP inom faktor B så avgör ovanstående algoritm ifall det finns en Hamiltoncykel i G vilket är NP-Fullständigt. Det ger motsägelse.

Kappsäcksproblemet

Indata är n objekt där objekt i beskrivs av paret (w_i, u_i). w_i är objektets vikt och u_i är dess nytta; båda är positiva heltal.
Beslutsproblemet är att avgöra om det går att välja ut objekt av total vikt högst W så att den totala nyttan blir minst U.
Detta problem är **NP**-fullständigt.
Motsvarande optimeringsproblem kan formuleras

$$\begin{aligned} &\max \sum_i x_i u_i \\ &\text{så att } \sum_i x_i w_i \leq W \\ &\qquad x_i \in \{0,1\}\, \forall i \end{aligned}$$

DynP-lösning av kappsäcksproblemet

Låt $V_{i,j}$ vara den minsta vikten som ger nyttan i när bara de j första objekten får väljas. Här är $1 \leq j \leq n$ och $0 \leq i \leq n \max u_i$ eftersom målfunktionen är högst $n \max u_i$.

Då är

$$
\begin{aligned}
&V_{0,j} = 0, \quad j = 1, \ldots, n \\
&V_{u_1,1} = w_1 \\
&V_{i,1} = \infty, \quad i \neq u_1 \\
&V_{i,j} = \min\bigl\{V_{i,j-1}, V_{i-u_j,j-1} + w_j\bigr\}
\end{aligned}
$$

Det tar $O(n^2 \max u_i)$ tid att fylla i hela tabellen.
Optimalvärdet är det största i sådant att
$V_{i,n} \leq W$.
Ur tabellen är det sedan lätt att hitta en mängd som ger detta värde på målfunktionen.

Pseudopolynomiska algoritmer

Algoritmen A är *pseudopolynomisk* om tidskomplexiteten är polynomisk i n, indatas längd, och M, det största talet som ingår i indata.
Obs:
En algoritm som är pseudopolynomisk kan trots allt vara exponentiell i n. Om indata är ett n-siffrigt binärt tal kan ju $M \sim 2^n$.
Ett **NP**-fullständigt problem som bara kan ha en pseudopolynomisk algoritm om $\mathbf{P} = \mathbf{NP}$ är *starkt **NP**-fullständigt.*

Approximationsalgoritm för kappsäcksproblemet

Den exakta algoritmen fungerar bra om nyttokoefficienterna u_i i indata är små. Skala därför dessa med en faktor δ och kör den exakta algoritmen:

```
ApproxKnapsack(U, W)
(1)   δ ← 2n / (ε u_max)
(2)   for i = 1 to n
(3)       u'_i ← ⌊δ u_i⌋
(4)   return ExactKnapsack(U', W)
```

(Vi låter $u_{max} = \max u_i$ och $w_{max} = \max w_i$.)
Frågor:

- Vilken approximationskvot fås?
- Vilken är tidskomplexiteten?

Analys av algoritmen

Vi antar att

1. $w_{max} \leq W$; $w_i > W$ är ointressant
2. $W \leq n w_{max}$; annars består lösningen av alla objekt

Låt *opt* vara optimum till (U, W)-instansen och *approx* vara målfunktionsvärdet hos den approximativa lösningen.
Låt vidare $I \subseteq [n]$ vara indexen till de objekt som ingår i lösningen till (U, W)-instansen och I' vara lösningen till (U', W)-instansen. Då är

$$\begin{aligned} approx &= \sum_{i \in I'} u_i \geq \frac{1}{\delta} \sum_{i \in I'} \lfloor \delta u_i \rfloor \geq \frac{1}{\delta} \sum_{i \in I} \lfloor \delta u_i \rfloor \\ &\geq \frac{1}{\delta} \big(\delta \sum_{i \in I} u_i - n \big) = \frac{1}{\delta} (\delta opt - n) = \\ &= opt - \frac{n}{\delta} \geq opt(1 - \epsilon/2) \end{aligned}$$

Analys, forts.

D.v.s. approximationskvoten är

$$\frac{opt}{approx} \leq \frac{1}{1 - \epsilon/2} \leq 1 + \epsilon$$

Körtiden är $O(n^3/\epsilon)$.
Det betyder att approximationskvoten kan fås godtyckligt nära 1 i polynomisk tid.
Detta är ett exempel på ett *approximationsschema* (eng. **P**olynomial **T**ime **A**pproximation **S**cheme, PTAS).
Flera PTAS:er för andra **NP**-fullständiga optimeringsproblem bygger på samma typ av skalning som i algoritmen ovan.

Terminologi

NPO
Klassen av optimeringsproblem i **NP**.

APX
De problem som kan approximeras inom någon konstant.

PTAS
De problem som kan approximeras inom varje konstant $1 + \epsilon$.

Det finns ett flertal problem (t.ex. minimal hörntäckning) som tillhör **APX** men inte **PTAS** om inte $\mathbf{P} = \mathbf{NP}$.
Det finns ett specialfall av Handelsresandeproblemet som går att approximera.

Approximation av TSP med triangelolikhet

Triangelolikheten: $w(i, j) \leq w(i, k) + w(k, j)$ för alla noder i, j, k.

Triangelolikheten medför att om $i, j, k_1, k_2, ..., k_s$ bildar en cykel i grafen så gäller $w(i, j) \leq w(i, k_s) + w(k_s, k_{s-1}) + ...w(k_1, j)$.

Problemet TSP med triangelolikhet brukar kallas Δ TSP.

Sats: Δ TSP är **NP**-fullständigt.

Antag att vi har ett minimalt spännande träd T i grafen. Om vi går fram och tillbaka från en nod i trädet till alla andra noder får vi en väg av längd $2w(T)$

där $w(T)$ är summan av kantvikterna i trädet. Vägen är förstås ingen tillåten lösning eftersom den inte är en cykel. Låt C vara en optimal cykel.

$w(C) = OPT$. Eftersom C är spännade träd + en kant fås $w(T) \leq w(C)$.

$$2 \cdot w(T) \leq 2 \cdot w(C) \leq 2 \cdot OPT$$

Vi kan göra om trädvandringen till en cykel C_1 genom att vandra runt noderna i den ordning de besöks i *inorderordningen* i trädet.

Påstående: $w(C_1) \leq 2 \cdot w(T)$

Det går att visa genom upprepad användning av triangelolikheten.

Vi har nu:

$$w(C) \leq w(C_1) \leq 2 \cdot w(T) \leq 2 \cdot w(C)$$

Vi sätter $APP = w(C_1)$. Vi får då:

$$OPT \leq APP \leq 2 \cdot OPT$$

Beräkningen av APP går att göra i polynomiell tid. Approximationskvoten blir $B = 2$.

Det finns en mer avancerad algoritm för approximation av Δ TSP som heter *Christofides algoritm*. Den bygger på samma idéer som ovanstående algoritm men lyckas pressa ner approximationskvoten till $\frac{3}{2}$.

Lådpackning

Givet tal $S = \{s_1, s_2, ..., s_n\}$ $0 < s_i \leq 1$ skall vi packa talen i lådor som bara rymmer summan 1. Vi skall använda *så få lådor som möjligt*

Beslutsproblem: Givet S och K, går det att packa talen i K lådor?

Problemet är **NP**-fullständigt.

Bevis: Vi visar att PARTITION $\leq$ LÅDPACKNING.

Givet en lista P med tal beräknar vi $W = \sum_i p_i$. Vi kan anta att alla $p_i \leq \frac{W}{2}$ (annars skulle svaret på partionsproblemet vara NEJ trivialt). Skala om:

$s_i = \frac{2p_i}{W}$

Sätt $K = 2$. Det ger vår instans av LÅDPACKNING.

Approximation av Lådpackning

Vi beskriver en metod som kallas First Fit Decreasing (FFD).

Sortera S så att $s_{\geq} s_2 \geq ... \geq s_n$. Vi tänker oss att vi har en följd av tomma lådor beredda. Vi placerar s_1 i första lådan. Sedan placeras talen i tur och ordning i den *första* lådan den får plats i.

Ex: $S = \{0,8\ ,\ 0,5\ ,\ 0,4\ ,\ 0,4\ ,\ 0,3\ ,\ 0,2\ ,\ 0,2\ ,\ 0,2\}$.

FFD placerar talen enligt:

L_1	0,8 , 0,2
L_2	0,5 , 0,4
L_3	0,4 , 0,3 , 0,2
L_4	0,2

Denna placering är inte optimal eftersom talen kan placeras i 3 lådor. (Hur?)

Komplexiteten blir $O(n^2)$. Vi gör nu en analys av hur bra approximationen är:

Påstående: Låt OPT vara det optimala (minimala) antalet lådor som krävs. Alla objekt som placeras i "överflödiga" lådor av algoritmen har storlek $\leq \frac{1}{2}$.

Varför?: Tal som är $> \frac{1}{2}$ måste hamna i olika lådor. Det finns inga val i det fallet. De enda tal som kan placeras "fel" är de som är $\leq \frac{1}{2}$.

Påstående: Antalet tal som placeras i "överflödiga" lådor är $\leq OPT - 1$.

Bevis: Vi vet att alla talen kan placeras i OPT antal lådor. Det betyder att $\sum_i s_i \leq OPT$.
Antag att OPT st objekt med storlek $t_1, t_2, ..., t_{OPT}$ placeras i överflödiga lådor. Låt b_i vara summan av innehållet i låda i för $1 \leq i \leq OPT$ efter att approximationsalgoritmen körts. Då måste $t_i + b_i > 1$.

$\sum_i s_i \geq \sum_{i=1}^{OPT} b_i + \sum_{i=1}^{OPT} t_i = \sum_{i=1}^{OPT} (b_i + t_i) > OPT$. Det ger motsägelse.

Om $OPT = m$ så placeras alltså högst $m - 1$ av storlek $\frac{1}{2}$ i överflödiga lådor.

$APP \leq m + \lceil \frac{m-1}{2} \rceil$

$B \leq 1 + \frac{\lceil \frac{m-1}{2} \rceil}{m} \leq 1 + \frac{\frac{m}{2}}{m} = \frac{3}{2}$.
Det betyder att algoritmen approximerar inom faktorn $\frac{3}{2}$.