5. Approximation algorithms

- **TSP med trekantsulighed**
- **Generel TSP**
- **Randomization**
- **FPTAS og KNAPSACK**

Mange NPC-problemer er for vigtige til at vi bare vil undlade at forsøge at løse dem, fordi vi ikke kan finde en polynomiel algoritme til det. Vi kan i stedet finde en approksimationsalgoritme, som garanterer at finde en løsning indenfor en vis ratio:

En polynomiel tids approksimationsalgoritme **har approksimationsratio på $\boldsymbol{\rho(n)}$** for et input n, hvis forholdet mellem costen af den optimale løsning C^* og costen af algoritmens løsning C er: $\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n)$. $\frac{C^*}{C}$ er ratioen for et maksimeringsproblem, $\frac{C}{C^*}$ er ratioen for et minimeringsproblem.

TSP med trekantsulighed: Graf $G = (V, E)$, hvor hver kan har en cost c. Vi skal finde en tur (ham-cycle) i G, så kort som mulig. Cost-funktionen opfylder trekantsuligheden: $c(u,w) \leq c(u,v) + c(v,w)$.

I følgende algoritme finder vi en lower-bound for den optimale tur, vha. et minimum-spanning-tree. Et MST er blot en TSP-tur med én kant fjernet.

```
APPROX-TSP-TOUR(G, c)
   select a vertex r ∈ V[G] to be a "root" vertex
   compute a minimum spanning tree T for G from root r
           using MST-PRIM(G, c, r)
   let L be the list of vertices visited in a preorder tree walk of T
   return the hamiltonian cycle H that visits the vertices in the order L
```

Ovenstående algoritme er en 2-approksimationsalgoritme, dvs. den garanterer at turen, den finder er maks. dobbelt så lang som den optimale.

Bevis: Algoritmen kører i polynomiel tid, så vi skal vise, at den har en approksimationsratio på 2.

Vi lader H^* være den optimale tur. MST T giver os en lower-bound for H^*: $\boldsymbol{c(T) \leq c(H^*)}$.

Vi har en full walk W, som er samtlige knuder, der besøges fra root og hele vejen rundt (inkl. dupl.). Denne må nødvendigvis have værdi $\boldsymbol{c(W) = 2c(T)}$.

Hvis vi kombinerer de to ovenstående: $\boldsymbol{c(W) \leq 2c(H^*)}$

Dog er W ikke en tur, da nogle knuder besøges flere gange. Nu sletter vi alle duplikater fra denne liste – længden forøges ikke pga. trekantsuligheden. Stien vi får ud af dette er en ham-cyclen H, som har costen $\boldsymbol{c(H) \leq c(W)}$, da vi får H ved at fjerne kanter fra W.

Til sidst kombinerer vi, hvad vi ved og får: $\boldsymbol{c(H) \leq c(W) \leq 2c(H^*)}$, og dermed er det vist.

Hvis vi nu fjerne kravet om trekantsuligheden, så kan vi ikke finde en effektiv ρ –approksimationsalgoritme for $\rho \geq 1$, med mindre $P = NP$.

Bevis: Vi beviser det ved modbevis, og antager vi har en effektiv ρ –approksalgoritme A. Vi skal vise, at vi kan løse ham-cycle problemer effektivt med denne algoritme.

$G = (V, E)$ er en instans af ham-cycle problemet. Vi laver en ny graf $G' = (V, E')$ med samme knuder, men med kanter imellem alle. For hver kant $(u,v) \in E'$ har vi cost-funktionen:

$$c(u,v) = \begin{cases} 1 & (u,v) \in E \\ \rho|V| + 1 & ellers \end{cases}$$

Vi kigger på TSP (G', c):

1. Hvis G har en ham-cycle H, så er der en tur i G', som har værdien $|V|$.

2. Hvis G ikke har en ham-cycle, så skal en tur i G' bruge en kant, der ikke er i E, og dermed får den værdien mindst: $(|V|-1)+(\rho|V|+1)=|V|+\rho|V|>\rho|V|$.

Der er altså et "gap" på mindst $|V|$ imellem de to ture. Vi har dermed vist, at algoritmen A giver en løsning på højst ρ gange den optimale løsning, hvis der er en ham-cycle. Hvis der ikke er, giver vi en løsning på mindst $\rho|V|$.

A kan altså svare effektivt på, om der er en ham-cycle i en graf ud fra længden på turen. Men da HamCycle $\in NPC$ skulle der altså gælde, at $P=NP$, hvilket vi antog ikke gælder for dette bevis.

Randomization er en teknik, man kan bruge, når man skal designe en approksimationsalgoritme. Her taler vi om C som den forventede cost af en løsning.

MAXE3SAT: 3CNF-formel med forskellige variabler i hver klausul. Hvor mange klausuler kan vi maksimalt tilfredsstille?

Sætning: Givet en instans af MAXE3SAT med n variabler og m klausuler, har vi en 8/7-randomiseret approksimationsalgoritme, som sætter hver variabel til 1 med ss ½ og til 0 med ss ½. Vi forventer at mindst $\frac{7m}{8}$ klausuler tilfredsstilles.

Bevis: Vi har uafhængigt sat variablerne til 0 eller 1 med de angivne ss. Vi definerer:

$$Y_i=\begin{cases}1 & klausul\ i\ er\ opfyldt\\ 0 & ellers\end{cases}\quad,\quad i=1,\dots,n$$

$\Pr[Y_i=0]=(½)^3=\frac{1}{8}$, alle tre literaler skal være false, da ingen variabler optræder flere gange.

$\Pr[Y_i=1]=1-\frac{1}{8}=\frac{7}{8}$, hvis klausulen ikke indeholder en variable og dens negering.

Den forventede værdi er $E[Y_i]=0\cdot\frac{1}{8}+1\cdot\frac{7}{8}=\frac{7}{8}$.

Y er en samling af alle Y_i'er:

$$E[Y]=E\left[\sum_{i=1}^{m}Y_i\right]=\sum_{i=1}^{m}E[Y_i]=\sum_{i=1}^{m}\frac{7}{8}=\frac{7m}{8}$$

Der kan maksimalt tilfredsstilles m klausuler, og dermed er approksimationsratioen $\frac{C^*}{C}\Rightarrow\frac{m}{7m/8}=\frac{8}{7}$

Et approksimations scheme er en approksimationsalgoritme, som tager et $\epsilon>0$ som input (udover problemet), således at algoritmen er en $(1\pm\epsilon)-$approksimationsalgoritme. **Polynomial-time approximation scheme PTAS** har en for et $\epsilon>0$ en køretid polynomielt på størrelsen af inputtet. Vi kan risikere, at køretiden forøges kraftigt, hvis ϵ formindskes. Vi siger at et approksimations scheme er **fully-PTAS**, hvis køretiden er polynomiel i både $\frac{1}{\epsilon}$ og inputstørrelsen.

Sætning:

1. Sæt $V=\max(v_i|w_i\leq W)$
2. $B=\frac{\epsilon V}{n}$
3. $v_i'=\left\lfloor\frac{v_i}{B}\right\rfloor$
4. Find optimal S' dynamisk programmeringsalgoritme for KNAPSACK med input $v_i{}'$ og w_i
5. Returner S'

Køretid: $O(n^2V)\geq O(n^2V')=O\left(n^2\frac{V}{B}\right)=O\left(n^2\frac{V}{\frac{\epsilon V}{n}}\right)=O\left(n^2\frac{Vn}{\epsilon V}\right)=O\left(n^3\frac{1}{\epsilon}\right)$