

3. januar

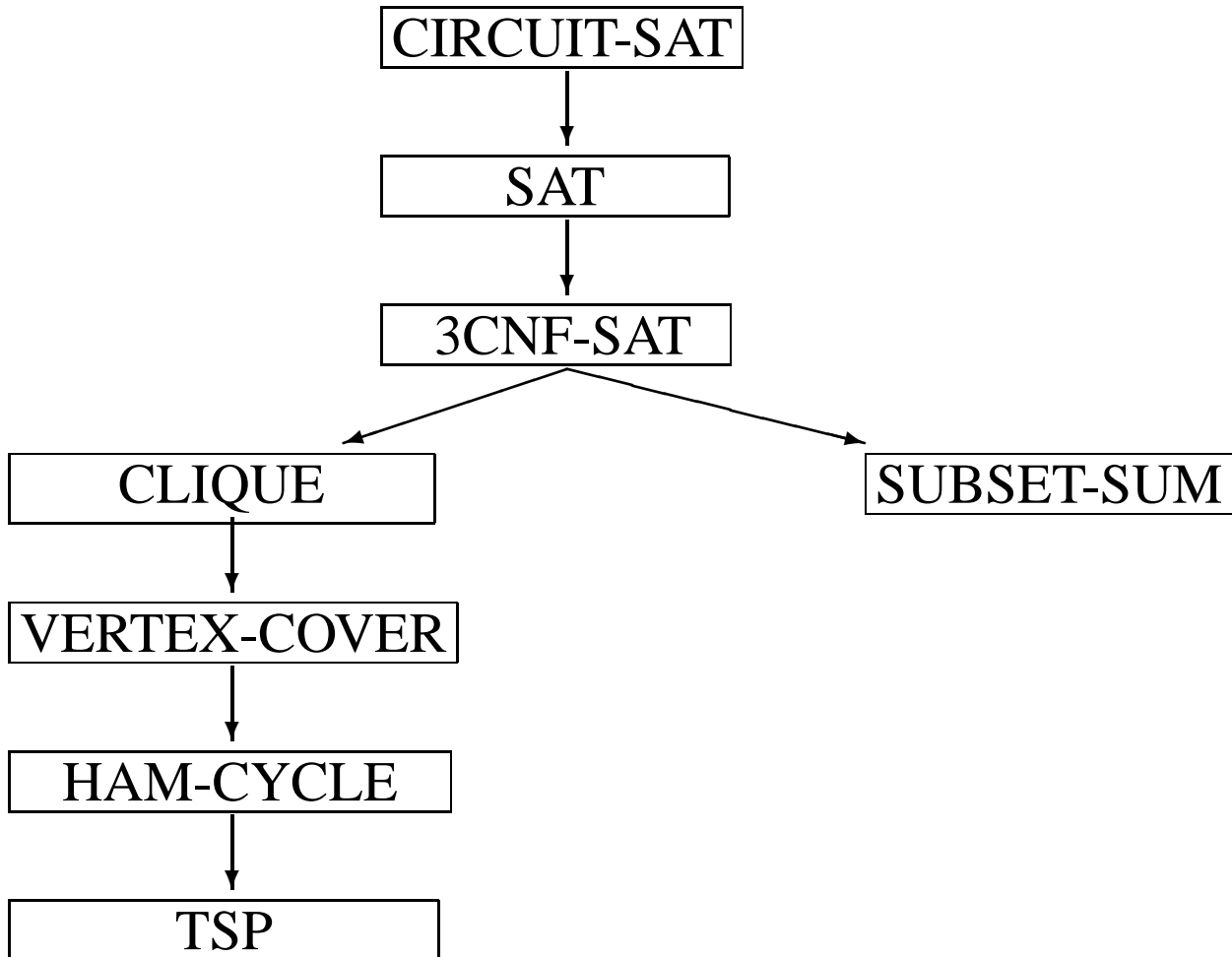
Resume sidste to gang

- Sprog L : mængden af instanser for et afgørlighedsproblem hvor svaret er 1.
- $\mathcal{P} = \{L : L \text{ genkendes af en algoritme i polynomiel tid}\}$
- $\mathcal{NP} = \{L : L \text{ verificeres af en polynomiel tids algoritme}\}$
- Q er \mathcal{NP} -fuldstændig hvis og kun hvis $Q \in \mathcal{NP}$ og $\forall R \in \mathcal{NP} : R \leq_{pol} Q$
- CIRCUIT-SAT er \mathcal{NP} -fuldstændig
- SAT er \mathcal{NP} -fuldstændig
- 3CNF-SAT er \mathcal{NP} -fuldstændig
- CLIQUE er \mathcal{NP} -fuldstændig
- VERTEX-COVER er \mathcal{NP} -fuldstændig

Problemer $Q \leq_{pol} R$ reduceres kun den ene vej, men f.eks. $CLIQUE \leq_{pol} CIRCUIT-SAT \leq_{pol} 3CNF-SAT$

Oversigt

Vi fortsætter beviser for \mathcal{NP} -fuldstændighed.



I dag:

- Hamilton kreds
- Traveling salesman problem
- Subset-sum problem

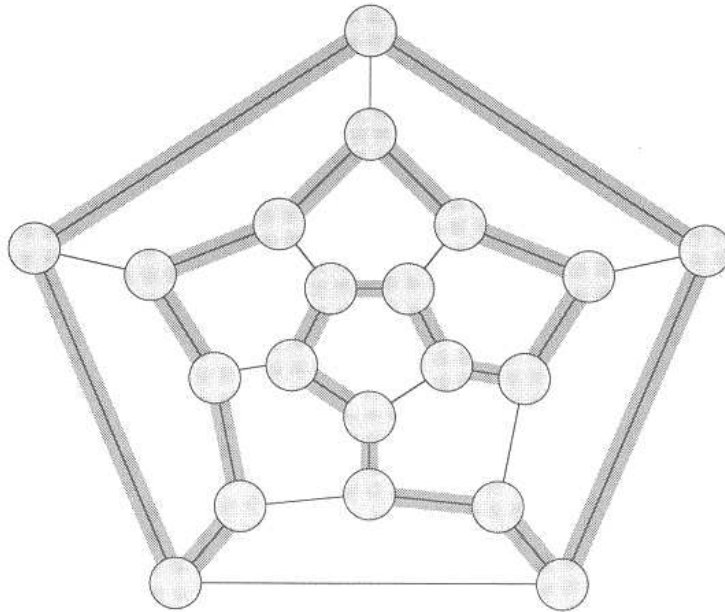
Start på branch-and-bound

Hamilton-kreds

HAM-CYCLE afgørlighedsproblem:

HAM-CYCLE = $\{ \langle G \rangle : \text{der findes en kreds i grafen } G = (V, E) \text{ s\aa hver knude bes\oeges een gang} \}$

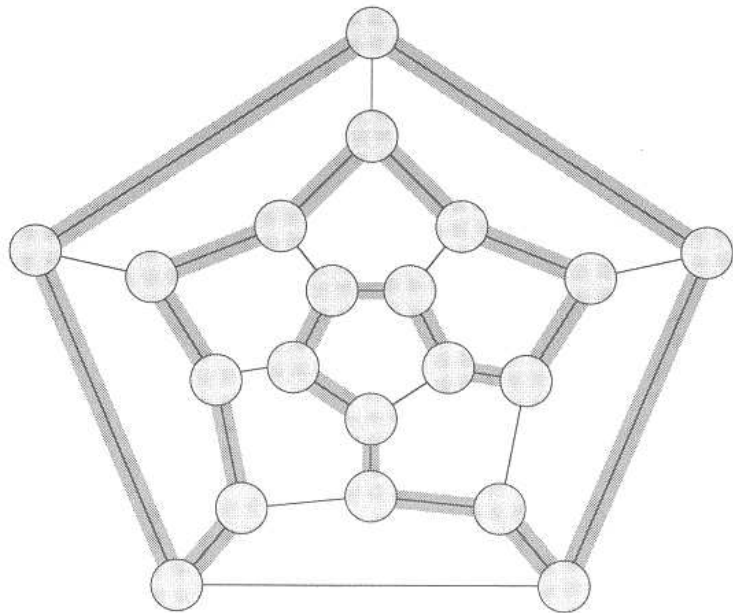
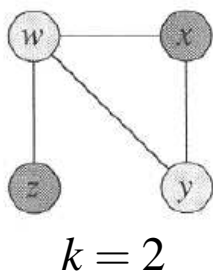
For eksempel



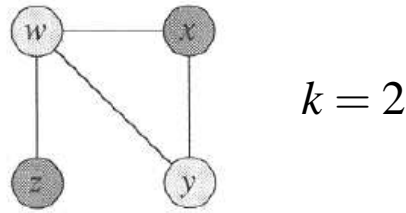
HAM-CYCLE er \mathcal{NP} -fuldstændig

- 1 Bevis at HAM-CYCLE $\in \mathcal{NP}$
- 2 Vælg et kendt \mathcal{NP} -fuldstændigt problem
- 3 Beskriv en algoritme f som afbilder
VERTEX-COVER \mapsto HAM-CYCLE
- 4 Bevis at f opfylder $x \in \text{VERTEX-COVER} \Leftrightarrow f(x) \in \text{HAM-CYCLE}$ for alle $x \in \{0, 1\}^*$
- 5 Bevis at f kører i polynomiel tid.

VERTEX-COVER og HAM-CYCLE



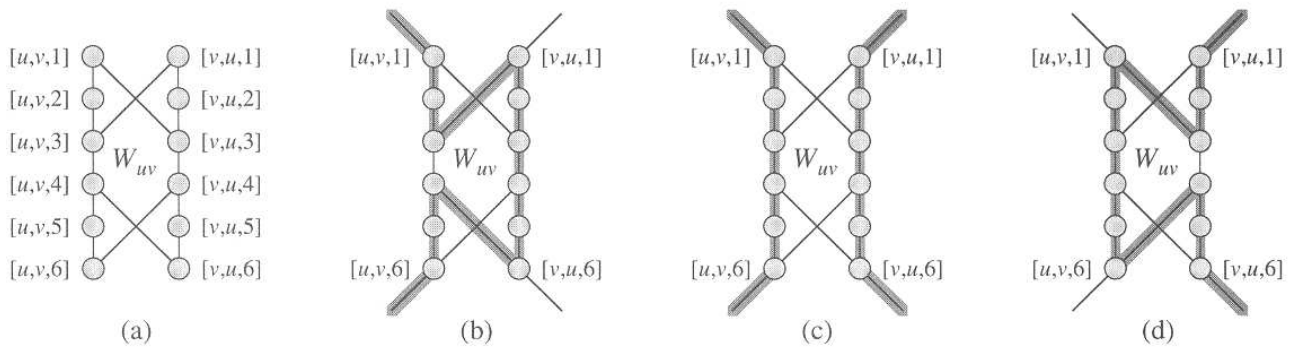
HAM-CYCLE er \mathcal{NP} -fuldstændig



Intuition: En knude-overdækning opfylder

- a) vælger k knuder
- b) når knude vælges, markerer vi insidente kanter
- c) alle kanter skal markeres 1 eller 2 gange

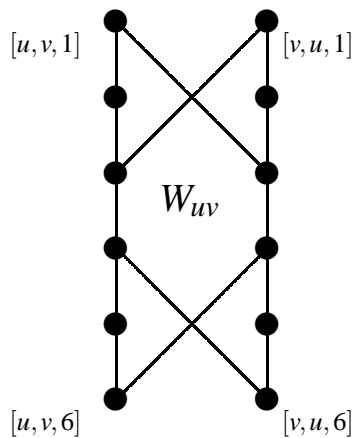
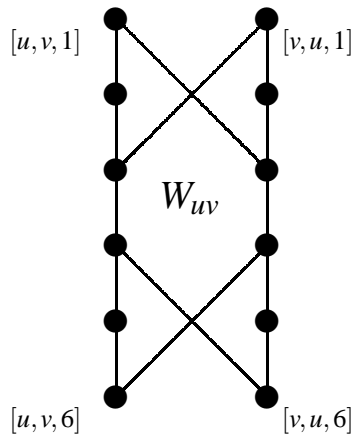
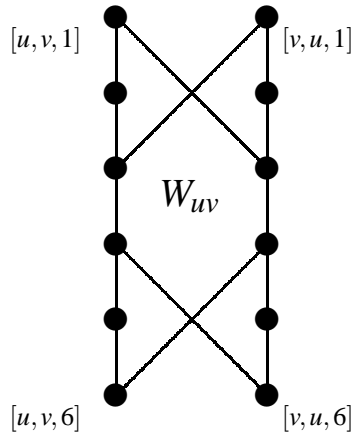
Hver kant repræsenteres af widget (dims)



- c) hver widget kan gennemløbes 1 eller 2 gange
- a) tilføjer k ekstra knuder s_1, s_2, \dots, s_k
- b) når besøger knude s_i gennemløbes insidente kanter

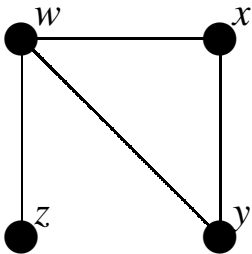
HAM-CYCLE er \mathcal{NP} -fuldstændig

Widget (dims, dippedut, dingnot)

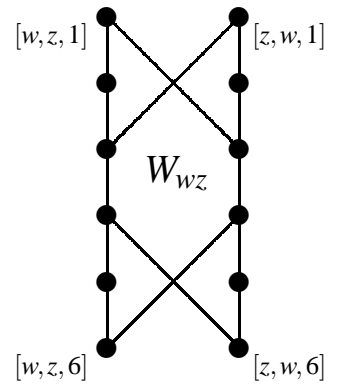
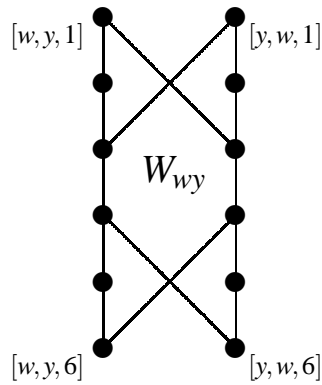
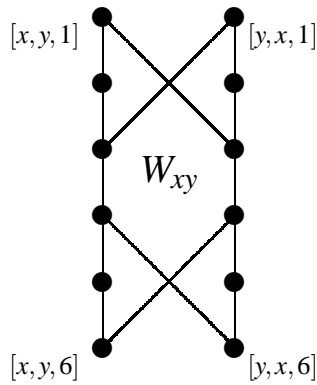
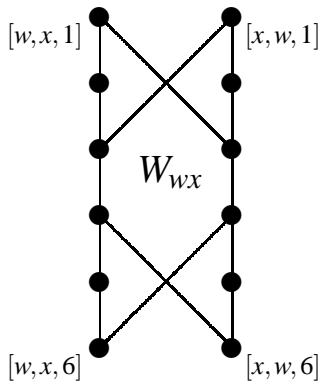


HAM-CYCLE er \mathcal{NP} -fuldstændig

Konstruktion

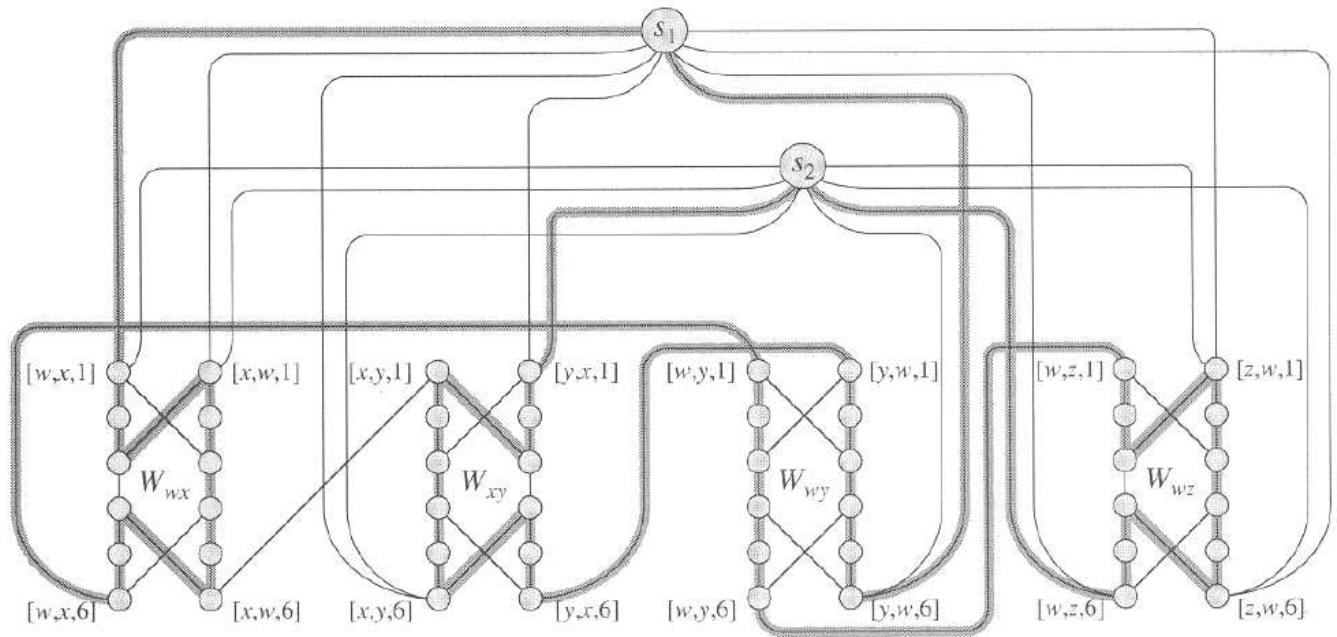
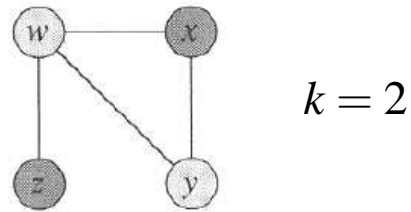


$$k = 2$$



HAM-CYCLE er \mathcal{NP} -fuldstændig

Samlet graf



HAM-CYCLE er \mathcal{NP} -fuldstændig

- f opfylder

$$x \in \text{VERTEX-COVER} \Rightarrow f(x) \in \text{HAM-CYCLE}$$

- f opfylder

$$f(x) \in \text{HAM-CYCLE} \Rightarrow x \in \text{VERTEX-COVER}$$

- f kører i polynomiel tid

– knuder:

* i widgets: $12E$

* knuder s_1, \dots, s_k : k

– kanter:

* i widgets: $14E$

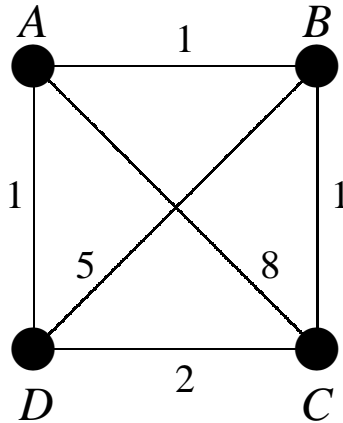
* knuder s_1, \dots, s_k til widgets: $2kV$

* imellem widgets: $2E - V$

– Bemærk: $k \leq V$

Traveling salesman problem

Givet en ikke-orienteret graf $G = (V, E)$ med vægte (afstande) c_{ij} knyttet til hver af kanterne e_{ij} . Find en Hamilton-kreds af mindste længde.



Afgørlighedsproblem

$TSP = \{ \langle G, c, k \rangle : G \text{ har en Hamilton-kreds af længde } \leq k \}$

hvor

$G = (V, E)$ er en komplet graf
 c er en afstands matrix
 k er et heltal

TSP er \mathcal{NP} -fuldstændig

- 1 Bevis at $\text{TSP} \in \mathcal{NP}$
- 2 Vælg et kendt \mathcal{NP} -fuldstændigt problem HAM-CYCLE
- 3 Beskriv en algoritme f som afbilder HAM-CYCLE \mapsto TSP
- 4 Bevis at f opfylder $x \in \text{HAM-CYCLE} \Leftrightarrow f(x) \in \text{TSP}$ for alle $x \in \{0, 1\}^*$
- 5 Bevis at f kører i polynomiel tid.

Reduktion:

- Givet en instans $G = (V, E)$ af HAM-CYCLE
- Konstuer komplet graf $G' = (V, E')$
- Afstande

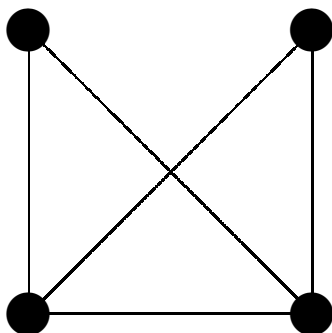
$$c_{ij} = \begin{cases} 0 & \text{hvis } (i, j) \in E \\ 1 & \text{hvis } (i, j) \notin E \end{cases}$$

- Sæt $k = 0$
- Ækvivalente TSP: $\langle G', c, k \rangle$

TSP er \mathcal{NP} -fuldstændig

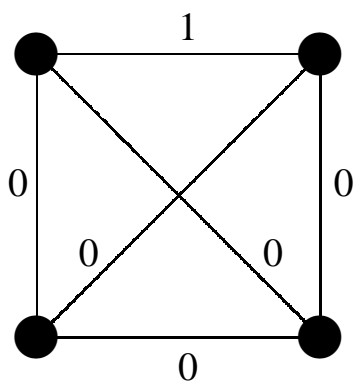
HAM-CYCLE

(V, E)



TSP

(V, E', c)



$k = 0$

Subset-sum problem

Subset-sum problemet (delmængde sum):

- Givet mængde af heltal $S = \{s_1, s_2, \dots, s_n\}$ samt heltal t
- Findes der en delmængde $S' \subseteq S$ så

$$\sum_{j \in S'} s_j = t$$

For eksempel

$S = \{1, 2, 7, 14, 49, 98, 343, 686, 2409, 2793, 16808, 17206, 117705, 117993\}$

$t = 138457$

har løsning

$$S' = \{1, 2, 7, 98, 343, 686, 2409, 17206, 117705\}$$

Afgørlighedsproblem

SUBSET-SUM = $\{ \langle S, t \rangle : \text{der eksisterer en delmængde } S' \subseteq S \text{ så } \sum_{j \in S'} s_j = t \}$

SUBSET-SUM er \mathcal{NP} -fuldstændig

Bevisets gang

- 1 Bevis at SUBSET-SUM $\in \mathcal{NP}$
- 2 Vælg et kendt \mathcal{NP} -fuldstændigt problem 3CNF-SAT.
- 3 Beskriv en algoritme f som afbilder 3CNF-SAT \mapsto SUBSET-SUM.
- 4 Bevis at f opfylder

$$x \in \text{3CNF-SAT} \Leftrightarrow f(x) \in \text{SUBSET-SUM}$$

for alle $x \in \{0, 1\}^*$

- 5 Bevis at f kører i polynomiel tid.

SUBSET-SUM er \mathcal{NP} -fuldstændig

3CNF-SAT

$$\begin{aligned}\phi &= C_1 \wedge C_2 \wedge C_3 \wedge C_4 \\ &= (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)\end{aligned}$$

tilfredsstillende tildeling: $x_1 = 0, x_2 = 0, x_3 = 1$

Ved at literaler i clausul er forskellige

Transformation

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
x_1	$v_1 =$	1	0	0	1	0	0	1
$\neg x_1$	$v'_1 =$	1	0	0	0	1	1	0
x_2	$v_2 =$	0	1	0	0	0	0	1
$\neg x_2$	$v'_2 =$	0	1	0	1	1	1	0
x_3	$v_3 =$	0	0	1	0	0	1	1
$\neg x_3$	$v'_3 =$	0	0	1	1	1	0	0
	$s_1 =$	0	0	0	1	0	0	0
	$s'_1 =$	0	0	0	2	0	0	0
	$s_2 =$	0	0	0	0	1	0	0
	$s'_2 =$	0	0	0	0	2	0	0
	$s_3 =$	0	0	0	0	0	1	0
	$s'_3 =$	0	0	0	0	0	2	0
	$s_4 =$	0	0	0	0	0	0	1
	$s'_4 =$	0	0	0	0	0	0	2
	$t =$	1	1	1	4	4	4	4

SUBSET-SUM er \mathcal{NP} -fuldstændig

formelt

- hver variabel x_i to heltal v_i og v'_i i S
- for hver klausul C_j to heltal s_j og s'_j i S
- målsum t har 1 i hver søjle svarende til variabel, 4 i hver søjle svarende til klausul
- regner 10-tals system (faktisk ville base 7 være nok)
- alle heltal i S er forskellige

SUBSET-SUM er \mathcal{NP} -fuldstændig

- viser

$$x \in 3\text{CNF-SAT} \Rightarrow f(x) \in \text{SUBSET-SUM}$$

- viser

$$x \in 3\text{CNF-SAT} \Leftarrow f(x) \in \text{SUBSET-SUM}$$

Kører i polynomiel tid

- S indeholder $2n + 2k$ heltal hvert heltal $n + k$ cifre
- output $(2n + 2k)(n + k)$ cifre
- hvert ciffer genereres i polynomiel tid

Opsummering

De \mathcal{NP} -fuldstændige problemer er vores bedste bud til at vise $\mathcal{NP} \neq \mathcal{P}$.

- Hvis der findes et \mathcal{NP} -fuldstændigt problem som er løseligt i polynomiel tid, så er $\mathcal{NP} = \mathcal{P}$.
- Hvis et problem i \mathcal{NP} ikke kan løses i polynomiel tid så kan ingen \mathcal{NP} -fuldstændige problemer løses i polynomiel tid.

Samlinger af \mathcal{NP} -fuldstændige problemer findes i

- Garey and Johnson (1979) "Computers and Intractability: a guide to the theory of NP-completeness".
- Crescenzi and Kann (1995) "A compendium of NP optimization problems".
(link findes på hjemmesiden)

Løsning af \mathcal{NP} -hårde problemer

- Løs til optimalitet i eksponentiel tid
- Find tilnærmet løsning i polynomiel tid

Oversigt

- Optimeringsproblemer \leftrightarrow afgørlighedsproblemer
- Klassen \mathcal{EXP} , og brute-force metoden
- Divide-and-conquer
- Grænseværdier
- Branch-and-bound
- Et konkret eksempel: knapsack problemet

Selv om alle \mathcal{NP} -fuldstændige problemer kan reduceres til hinanden skal specifik struktur udnyttes

Afgørlighedsproblem versus optimeringsproblem

Traveling Salesman, afgørlighedsproblem

$\text{TSP} = \{ \langle G, c, k \rangle : G = (V, E) \text{ er en komplet graf,} \\ c \text{ er en afstands matrix,} \\ k \text{ er et heltal,} \\ G \text{ har Hamilton-kreds af længde } \leq k \}$

Traveling Salesman, optimeringsproblem

$\text{TSP-OPT} = \{ \langle G, c \rangle : G = (V, E) \text{ er en komplet graf,} \\ c \text{ er en afstands matrix,} \\ \text{find korteste Hamilton-kreds} \}$

Optimeringsproblemer mere naturlige, “nemmere” at løse

\mathcal{NP} -hård

- Optimeringsproblem er \mathcal{NP} -hårdt \Leftrightarrow
- Tilhørende afgørlighedsproblem er \mathcal{NP} -fuldstændigt

Optimeringsproblemer og afgørlighedsproblemer

Optimeringsproblem i *maksimeringsform*

$$z = \max\{f(x) \mid x \in S\}$$

- $f(x)$ er objektfunktionen
- S er løsningsrummet
- z er optimale løsningsværdi
- optimale løsning, dvs. det x^* hvor $z = f(x^*)$
- verifikation af x^* lige så svært som at løse problem

Optimeringsproblem i *minimeringsform*

$$z = \min\{f(x) \mid x \in S\}$$

omformes ved at maksimere $-f(x)$

Knapsack problem

- *profit* af genstand $p_j \in \mathbb{N}$
- *vægt* af genstand $w_j \in \mathbb{N}$
- *kapacitet* af rygsæk $c \in \mathbb{N}$

$$z = \max \left\{ \sum_{j=1}^n p_j x_j \mid \sum_{j=1}^n w_j x_j \leq c, x_j \in \{0, 1\} \right\}$$

- objektfunktion $f(x) = \sum_{j=1}^n p_j x_j$
- løsningsrum $S = \{x \in \{0, 1\} \mid \sum_{j=1}^n w_j x_j \leq c\}$

Eksempel

$$c = 9, n = 7$$

j	1	2	3	4	5	6	7
p_j	6	5	8	9	6	7	3
w_j	2	3	6	7	5	9	4

Optimale løsning: vælg genstande 1 og 4, $z = 15$.

Søgebaserede algoritmer

- polynomielle problemer: konstruktiv algoritme
- \mathcal{NP} -hårde problemer: søgebaseret algoritme

Branch-and-bound

- Søgebaseret
- Paradigme
- Total gennemsøgning af løsningsrummet (brute-force)
- Matematiske overvejelser udelukker dele af løsningsrummet (grænseværditest)
- Opdeler løsningsrum (divide-and-conquer)
- “Samler” løsninger fra delproblemer

Brute-force metoder

Definition 1 Klassen $\mathcal{EX}\mathcal{P}$: afgørlighedsproblemer som kan løses i eksponentiel tid på deterministisk TM

$L \in \mathcal{EX}\mathcal{P} \Leftrightarrow \exists p(n) : \forall x, |x| = n, x$ kan afgøres i tid $2^{p(n)}$

For ethvert \mathcal{NP} -problem

- findes et “kort” (dvs. polynomielt) *certifikat*
- kan et certifikat *verificeres* i polynomiell tid

Sætning 1 Hvis $L \in \mathcal{NP}$ så gælder også at $L \in \mathcal{EX}\mathcal{P}$

Algoritme:

- Gennemløb alle kombinationer af certifikatet
- For hvert certifikat test om verifikationsalgoritmen returnerer “ja”

Køretid:

- Verifikationsalgoritme $A(x, y)$ hvor $|y| \leq p_2(|x|)$, køretid $p_1(|x|)$
- Alle kombinationer af certifikat $2^{|y|} \leq 2^{p_2(|x|)}$
- Hvert certifikat verificeres i $p_1(|x|)$
- Tid: $2^{p_2(|x|)} \cdot p_1(|x|) = 2^{p_2(|x|)} \cdot 2^{\log p_1(|x|)} = 2^{p_2(|x|) + \log p_1(|x|)}$

Løsningsrummet for et \mathcal{NP} -hårdt problem er begrænset!

Divide and Conquer

- Brute-force metoden for afgørlichedsproblemer
- Divide and conquer for optimeringsproblemer

$$z = \max\{f(x) \mid x \in S\}$$

Sætning 2

- $S = S_1 \cup \dots \cup S_k$ opdeling af S i mindre mængder
- $z_i = \max\{f(x) \mid x \in S_i\}$ løsningsværdi for S_i
- så gælder

$$z = \max_{i=1,\dots,k} z_i$$

Bevis:

$$\begin{aligned} z &= \max\{f(x) \mid x \in S\} \\ &= \max\{f(x) \mid x \in S_1 \cup \dots \cup S_k\} \\ &= \max_{i=1,\dots,k} \max\{f(x) \mid x \in S_i\} \\ &= \max_{i=1,\dots,k} z_i \end{aligned}$$

□

Delproblem: optimeringsproblem begrænset til S_i

Divide and Conquer, eksempel

Knapsack problem, $n = 3$ genstande, kapacitet $c = 6$

j	1	2	3
p_j	6	8	3
w_j	2	6	4

- Løsningsrum

$$S = \{(x_1, \dots, x_n) \mid \sum_{j=1}^n w_j x_j \leq c, x_j \in \{0, 1\}\}$$

- Opdeling af S : x_1 sættes til 0 eller 1

$$S_1 = \{(x_1, \dots, x_n) \subseteq S \mid x_1 = 0\}$$

$$S_2 = \{(x_1, \dots, x_n) \subseteq S \mid x_1 = 1\}$$

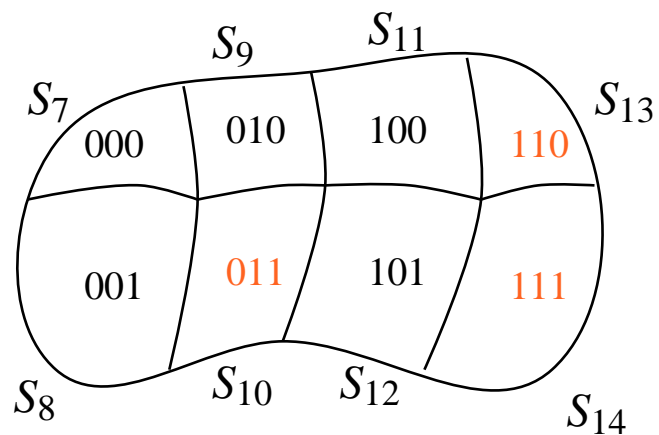
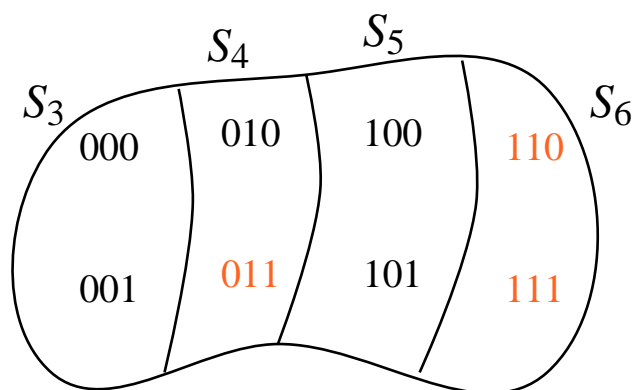
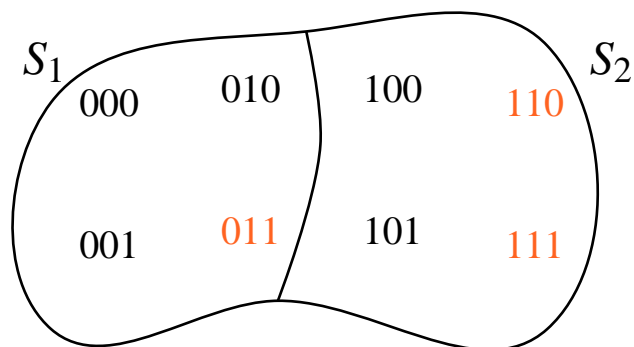
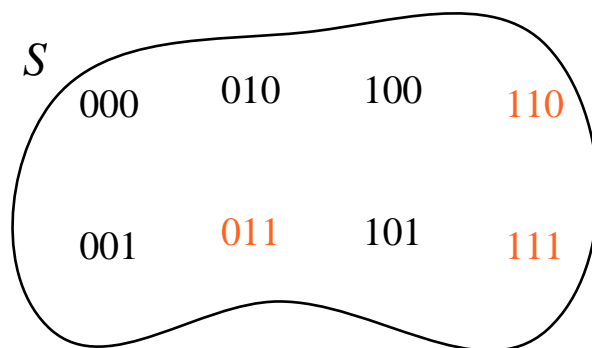
- Opdeling af S_1 : x_2 sættes til 0 eller 1

$$S_3 = \{(x_1, \dots, x_n) \subseteq S_1 \mid x_1 = 0, x_2 = 0\}$$

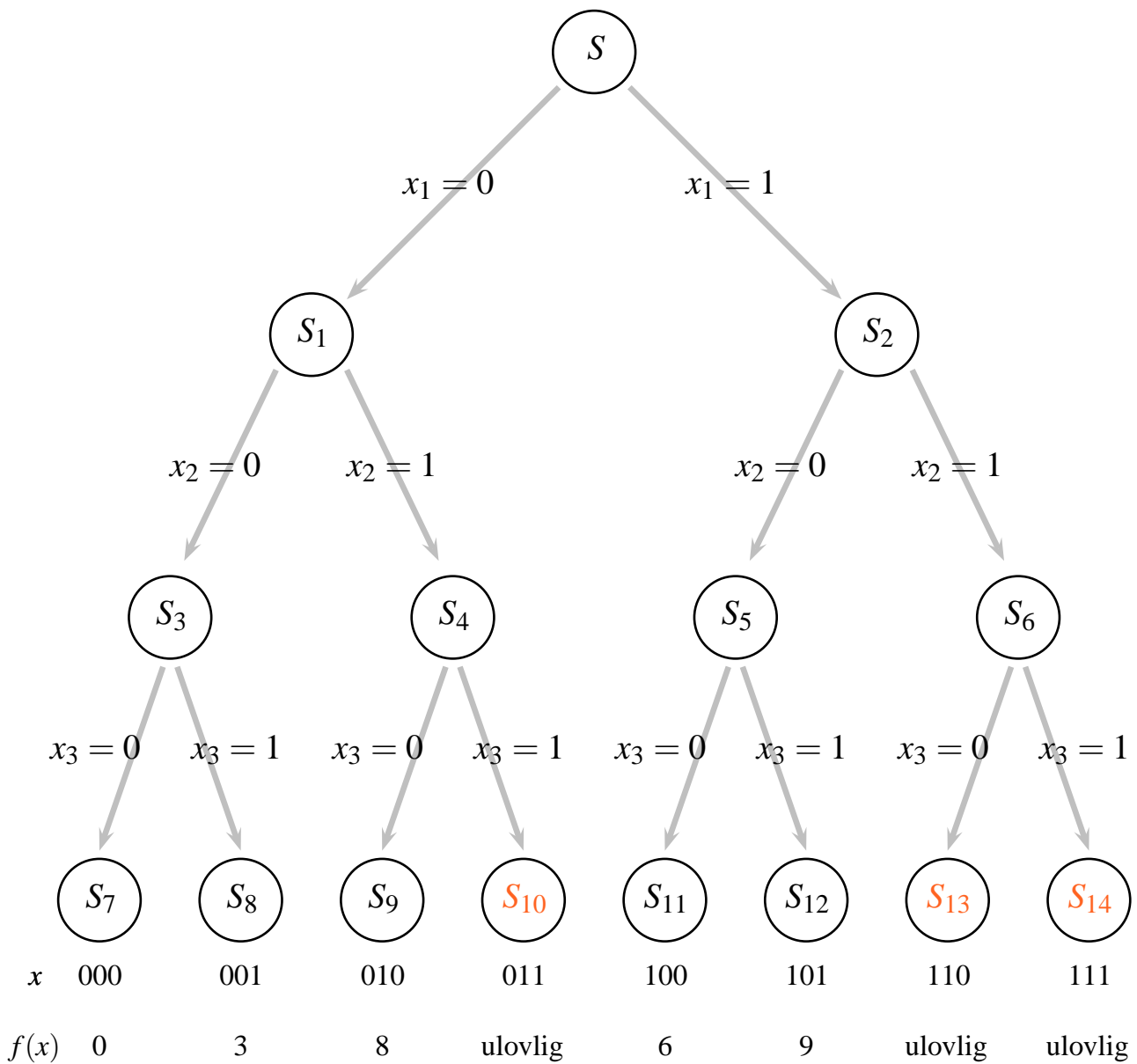
$$S_4 = \{(x_1, \dots, x_n) \subseteq S_1 \mid x_1 = 0, x_2 = 1\}$$

- Illustrere opdelingen af S med et søgetræ
- Hver knude svarer til et *delproblem* S_i

Optimal løsning til f.x. S_5 : $\max\{z_{11}, z_{12}\} = \max\{6, 9\} = 9$



Figur 1: Opdeling af løsningsrum. De lyse tal svarer til ulovlige løsninger, dvs. løsningsvektorer x hvor $\sum_{j=1}^n w_j x_j > c$.



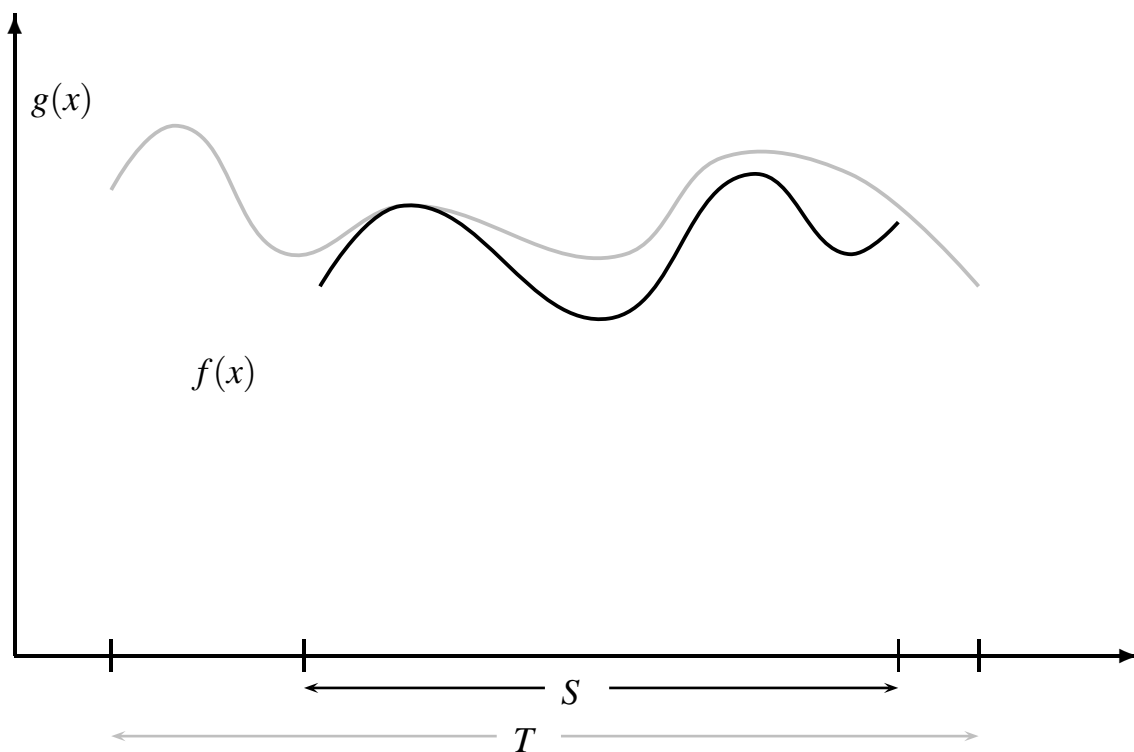
Figur 2: Søgetræ for knapsack problem. De lyse tal svarer til ulovlige løsninger, dvs. løsningsvektorer x hvor $\sum_{j=1}^n w_j x_j > c$.

Grænseværdier

$$z = \max\{f(x) \mid x \in S\} \quad \text{f.x. } S \text{ delproblem}$$

- *nedre grænseværdi* $\mathcal{L} \in \mathbb{R}: \mathcal{L} \leq z$
- *øvre grænseværdi* $\mathcal{U} \in \mathbb{R}: \mathcal{U} \geq z$

For enhver løsning $x' \in S$ er $f(x')$ nedre grænseværdi



Definition 2 $P : z = \max\{f(x) \mid x \in S\}$
 $R : z_R = \max\{g(x) \mid x \in T\}$

R er en relaxering af P hvis

(i) $S \subseteq T$

(ii) $g(x) \geq f(x)$ for alle $x \in S$

Sætning 3 Hvis R er en relaxering af P så gælder $z_R \geq z$.

Bevis

- x^* opt. løsning for P : $f(x^*) = z$
- Fra (ii) da $x^* \in S$ gælder $g(x^*) \geq f(x^*)$
- Fra (i) gælder $x^* \in T$
- $z_R = \max\{g(x) \mid x \in T\} \geq g(x^*) \geq f(x^*) = z$

Grænseværdier

- Løsning z_R til relaxering R er øvre grænseværdi \mathcal{U}
- Ofte kræves at relaxering kan løses effektivt

Grænseværditest

Lad \mathcal{U} være øvre grænseværdi for S_i

Hvis $\mathcal{U} \leq z$ så findes ikke forbedrende løsning i S_i

Divide and conquer

$S = S_1 \cup \dots \cup S_k$ opdeling af S

$$z_i = \max\{f(x) \mid x \in S_i\}$$

Sætning 4

- \mathcal{L}_i er nedre grænseværdi for delmængde S_i
- da er $\mathcal{L} = \max_{i=1, \dots, k} \mathcal{L}_i$ en nedre grænseværdi for S

Bevis Da $\mathcal{L}_i \leq z_i$ har vi $\mathcal{L} = \max_{i=1, \dots, k} \mathcal{L}_i \leq \max_{i=1, \dots, k} z_i = z$

Sætning 5

- \mathcal{U}_i er øvre grænseværdi for delmængde S_i
- da er $\mathcal{U} = \max_{i=1, \dots, k} \mathcal{U}_i$ en øvre grænseværdi for S

Bevis Da $z_i \leq \mathcal{U}_i$ har vi $\mathcal{U} = \max_{i=1, \dots, k} \mathcal{U}_i \geq \max_{i=1, \dots, k} z_i = z$

Sætning 6 $f(x)$ heltallig for ethvert $x \in S$

- \mathcal{U} er en øvre grænseværdi
- da er også $\lfloor \mathcal{U} \rfloor$ en øvre grænseværdi

Knapsack problem, øvre grænseværdi

Fraktionelle knapsack problem i Cormen

$$u_{\text{KP}}^1 = \max \left\{ \sum_{j=1}^n p_j x_j \mid \sum_{j=1}^n w_j x_j \leq c, 0 \leq x_j \leq 1 \right\}$$

Originale knapsack problem

$$f(x) = \sum_{j=1}^n p_j x_j$$

$$S = \left\{ (x_1, \dots, x_n) \mid \sum_{j=1}^n w_j x_j \leq c, x_j \in \{0, 1\} \right\}$$

Fraktionelle knapsack problem

$$g(x) = \sum_{j=1}^n p_j x_j$$

$$T = \left\{ (x_1, \dots, x_n) \mid \sum_{j=1}^n w_j x_j \leq c, 0 \leq x_j \leq 1 \right\}$$

Da $f(x) = g(x)$ er kriterium (ii) i definition 2 overholdt. Envidere er $S \subseteq T$, hvorfor kriterium (i) også er opfyldt.

Branch-and-bound

$$z = \max\{f(x) \mid x \in S\}$$

```
1   $\mathcal{L} := -\infty; L := \{S\}$ 
2  while  $L \neq \emptyset$ 
3    vælg et delproblem  $S_i$  fra  $L$ 
4     $L := L \setminus \{S_i\}$ 
5    if  $S_i \neq \emptyset$  then
6      find en øvre grænseværdi  $\mathcal{U}(S_i)$ 
7      find en lovlig løsning  $x$  i  $S_i$ 
8      if  $\mathcal{U}(S_i) > \mathcal{L}$  then
9        if  $f(x) > \mathcal{L}$  then  $\mathcal{L} := f(x); x^* := x$ 
10       opdel  $S_i$  i delproblemer  $S_i^1, \dots, S_i^k$ 
11       tilføj delproblemerne til  $L$ , dvs. sæt
            $L := L \cup \{S_i^1, \dots, S_i^k\}$ 
12     endif
13   endif
14 endwhile
```

- L en liste af delproblemer S_i givet ved de tilhørende løsningsrum. Listen L vil ofte være organiseret som en prioritetskø.
- Vi kalder delproblemerne i L for *åbne delproblemer*, mens delproblemer der allerede er blevet behandlet kaldes *lukkede delproblemer*.
- global nedre grænseværdi \mathcal{L} , tilhørende løsning x^* .

Branch-and-bound

En branch-and-bound algoritme for et maksimeringsproblem vil derfor bestå af følgende fire komponenter:

1. En *øvre grænseværdifunktion* som for et givet delproblem returnerer en øvre grænse på værdien af den bedste løsning vi kan finde i delrummet.
2. En *nedre grænseværdi* som på ethvert tidspunkt angiver den hidtil bedst kendte løsning.
3. En *søgestrategi* som fastlægger en rækkefølge for behandlingen af delproblemer.
4. En *forgreningsregel* som for alle delproblemer som ikke kan forkastes af grænseværditesten, angiver hvorledes det tilhørende løsningsrum skal opdeles. Dermed skabes to eller flere nye delproblemer.