

I dag

Løsning af \mathcal{NP} -hårde problemer

- Introduktion til Branch-and-bound
- Et konkret eksempel: knapsack problemet
- Mere om Branch-and-bound
 - Søgestrategi
 - Valg af grænseværdi funktion
 - Ivrig eller doven evaluering af grænseværdi
 - Forgreningsregler
 - Startløsning

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

Konkrete eksempler (næste gang)

Løsning af \mathcal{NP} -hårde optimeringsproblemer

Optimeringsproblemer

- Mere naturligt
- Nemmere at løse

Traveling Salesman, afgørlighedsproblem

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

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

Optimeringsproblemer

- Løsning kan ikke verificeres i polynomiel tid.
- Problem er ” \mathcal{NP} -hårdt”.
- Tilhørende afgørlighedsproblem er \mathcal{NP} -fuldstændigt.

Løsning af \mathcal{NP} -hårde problemer

Alternativer

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

\mathcal{NP} -fuldstændige problemer kan løses i eksponentiel tid

- Certifikat polynomielt langt
- Certifikat kan evalueres i polynomielt tid
- Kan derfor prøve alle mulige certifikater

Branch-and-bound paradigmet

Alternative navne:

- Branch-and-bound
- Del-og-hersk
- Implicit enumerering

Karakteristika:

- Systematisk gennemsøgning af alle mulige løsninger
- Dele af gennemsøgningen undgås pga. grænseværdi
- I værste fald prøves alle kombinationsmuligheder

Opdeling af løsningsrum

Løsningsrum deles op i hver iteration indtil hver mængde kun indeholder en enkelt mulig løsning.

- Opdeling kan ske ved at tildele en variabel værdier
- Opdeling må godt have overlappende delløsninger
- Opdeling må ikke udelukke lovlige løsninger

Branch-and-bound komponenter

Minimeringsproblem:

$$\min_{x \in S} f(x)$$

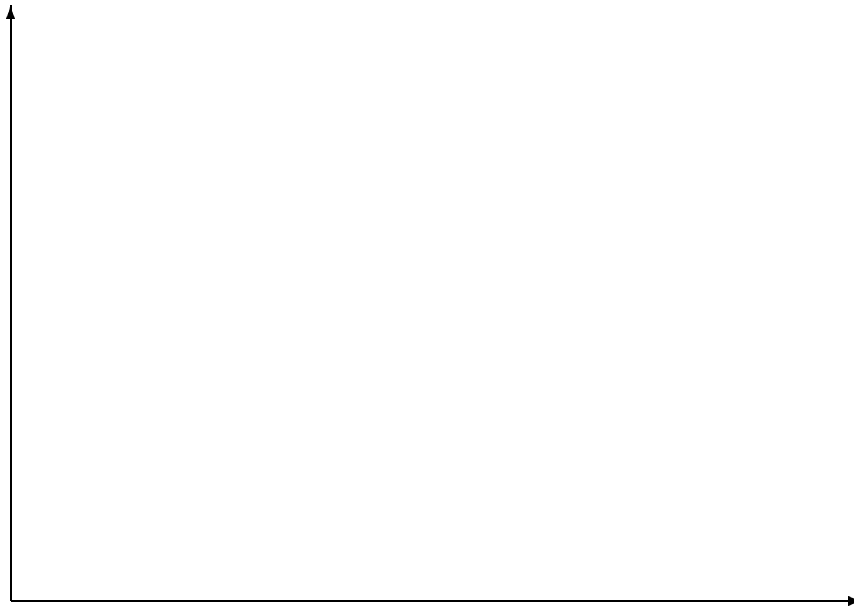
- $f(x)$ objektfunktion
 - S løsningsmængde
 - hidtil bedste løsning z (incumbent)
 - forgrening (branch) opdeler S i mindre dele S_i
-
- grænseværdi $\ell_i \leq f(x)$ for alle $x \in S_i$
 - forkastning af delløsning S_i (fathoming) hvis $\ell_i \geq z$

Grænseværdi

Nedre grænse $\ell_i \leq f(x)$ for $x \in S_i$.

Eksakt bestemmelse af ℓ_i tager lang tid

- Objektfunktion ændres
- Relaxering af begrænsninger (løsningsrum udvides)
- Kombination af begge



Hvis $g(x) \leq f(x)$ for alle $x \in S_i$ så

$$\min_{x \in S_i} g(x) \leq \min_{x \in S_i} f(x)$$

Hvis endvidere $S_i \subseteq T_i$ så

$$\min_{x \in T_i} g(x) \leq \min_{x \in S_i} g(x)$$

Lad $\ell_i = \min_{x \in T_i} g(x)$. Hvis $\ell_i \geq z$ så

$$z \leq \ell_i \leq f(x) \text{ for alle } x \in S_i$$

Knapsack Problem

Givet n genstande og een rygsæk

- Genstand j vejer w_j
- Nytteværdi af genstand j er p_j
- Rygsækkens indhold må højst veje c

Maksimeringsproblem

$$\begin{aligned} &\text{maximize} && \sum_{j=1}^n p_j x_j \\ &\text{subject to} && \sum_{j=1}^n w_j x_j \leq c \\ &&& x_j \in \{0, 1\}, \quad j = 1, \dots, n. \end{aligned}$$

Vigtigt problem

- Budgettering
- Transport
- Underproblem i andre algoritmer

Design af branch-and-bound algoritmer

- Søgestrategi
- Hvornår skal grænseværdi udregnes (ivrig/doven)
- Grænseværdi funktion
- Forgreningsregler
- Startløsning

Valg er problemspecifikke.

Søgestrategier

- Breadth first

- Depth first

- Best first

Søgestrategier

Breadth first

- Exponentielt antal knuder
- Måske fordel at alle knuder på samme niveau
- (Lige mange variable tildelt værdi, grænseværdi ligger)

Depth first

- Hvis træets dybde er n så $O(n)$ plads
- Stiler ikke nødvendigvis mod optimal løsning

Best first

- Exponentielt antal knuder
- Datastruktur til at organisere knuder efter grænseværdi
- Stiler mod optimal løsning

Hvornår skal grænseværdi udregnes

To strategier: Ivrig (*Eager*) og Doven (*Lazy*)

Datastruktur

$$P = \{(P_1, \ell_1), (P_2, \ell_2), \dots, (P_m, \ell_m)\}$$

liste, stak, prioritetskø (minimum)

Initielt $P_1 =$ hele løsningsrummet

Eager:

choose live node (P, ℓ)

subdivide P (branch)

for each new P_i

 fathom if not feasible

 update z if improved

 derive bound ℓ_i

if $\ell_i < z$ **then**

 add (P_i, ℓ_i) to nodes

else

 fathom P_i

Lazy:

choose live node (P, ℓ)

fathom if not feasible

derive bound ℓ

if $\ell < z$ **then**

 update z if improved

 subdivide P (branch)

for each new P_i :

 add (P_i, ℓ) to nodes

else

 fathom P_i

(minimeringsproblem)

Ivrig eller doven

Ivrig

- “Korrekt” grænseværdi for alle knuder
- Hvis z forbedres og knuder forkastes, er grænsev. beregningen spildt

Doven

- Forældreknudernes grænsværdi benyttes
- Hvis z forbedres og knude kan forkastes på basis af forældreknudes grænsværdi så sparet grænsev.beregning

Forskelle

- Da grænsværdi typisk kan beregnes i polynomiell tid, kun polynomiell faktor i forskel på køretid.
- Hvis z^* kendt fra begyndelsen, ingen forskel
- Stor forskel hvis parallel løsning

Opdeling af løsningsrum

Branching opdeler løsningsrum

Tilføjer ekstra begrænsning til hver mængde

Typisk ved at tildele en eller flere variable en værdi

- Dichotomisk

- Polytomisk

For et minimeringsproblem vil grænsværdi være voksende jo dybere ned i søgetræet vi kommer:

$$l_i \geq l_j \quad \text{hvis } i \text{ er afkom af } j$$

Hvor mange knuder skal besøges

- Lad z være nuværende løsning
- Lad z^* være optimal løsning
- *Kritiske delproblemer S_i*

$$l_i < z^*$$

- *Semi-kritiske delproblemer S_i*

$$l_i \leq z^*$$

Hvis z^* kendt fra starten skal alle kritiske knuder besøges uanset søgestrategi (bredde-først, dybde-først, bedste-først).

Hvis z^* først findes undervejs i forløbet risikerer vi også at besøge knuder med

$$l_i < z \quad \text{men} \quad l_i \geq z^*$$

Indsigt

Vigtigste valg i branch-and-bound algoritme

- grænseværdi funktion
- opspaltning i delproblemer
- god startløsning

Er disse valgt, skal alle kritiske delproblemer besøges for at bevise optimalitet

Sekundære valg:

- Søgestrategi (bedste-først, dybde-først, bredde-først) da primært mål at lede mod optimal løsning
- Ivrig eller doven evaluering af grænseværdi

Grænseværdi beregning

Relaxeringer

- ”Smid nogle krav væk”
- Udvid mængden af lovlige løsninger
- Tilstræb nemt (polynomielt) resterende problem

Hvilke begrænsninger skal relaxeres

- ”de grimme”
- tilbageværende problem er polynomielt løseligt (e.g. min spanning tree, assignment problem, linear programming)
- resterende problem er \mathcal{NP} -hårdt men gode teknikker findes (e.g. knapsack)
- begrænsninger som er svære at beskrive matematisk (e.g. cutting)
- begrænsninger som er for omfattende at beskrive explicit (e.g. subtour elimination in TSP)

Startløsning

Hvis kender optimale løsning z^* fra starten, så besøger breadth-first, depth-first, og best-first samme antal knuder.

- Konstruer løsning ved grådig heurstik
- Problemspecifik algoritme
- Approximationsalgoritme
- Metaheuristikker

Det kan normalt betale sig at basere søgestrategi på grådige principper.

Opsummering

Løsning af \mathcal{NP} -hårde optimeringsproblemer

- Optimeringsproblemer \leftrightarrow afgørlighedsproblemer
- \mathcal{NP} -hårdhed for optimeringsproblemer
- Optimeringsproblemer kan ikke verificeres i polynomielt tid
- Løs til optimalitet i eksponentiel tid
- Find tilnærmet løsning i polynomielt tid

Branch-and-bound

- Systematisk gennemsøgning af alle mulige løsninger
- Dele af gennemsøgningen undgås
- Der findes instanser som kræver at et eksponentielt antal delløsninger undersøges

Opsummering

Minimeringsproblem:

$$\min_{x \in S} f(x)$$

- $f(x)$ objektfunktion
- S løsningsmængde
- hidtil bedste løsning z (incumbent)
- forgrening (branch)
 - dichotomisk eller polytomisk opdeling
 - bredde-først, dybde-først, bedste-først søgning
- grænseværdi ℓ_i
 - overholder $\ell_i \leq f(x)$ for alle $x \in S_i$.
 - delløsning S_i forkastes (fathoming) hvis $\ell_i \geq z$.
 - ivrig eller doven evaluering af grænseværdi.
 - grænseværdi $\ell_i \geq \ell_j$ når i er afkom af j .
- startløsning
 - skal være så god som mulig: heuristikker
 - hvis $z = z^*$ så skal kun kritiske knuder besøges