INF-MAT-5380

http://www.uio.no/studier/emner/matnat/ifi/INF-MAT5380/

Leksjon 2

Leksjon 1: Oppsummering

- Kursinformasjon
- Motivasjon
  - Operasjonsanalyse
  - Kunstig intelligens
- Optimeringsproblemer (diskrete)
  - Matematisk program
  - COP
- Definisjon DOP
- Anvendelser
- Kompleksitetsteori
- Eksakte metoder, approksimasjonsmetoder
- Heuristikker
- Skisse av lokalsøk

Leksjon 2

- Eksempler på DOP
- Mer om lokalsøk
- Begrepsdefinisjoner
- Hovedproblem i lokalsøk

Eksempel på DOP: TSPTW

- Handelsreisende-problemet med tidsvinduer (TSPTW)
- Komplett graf med n noder (byer) $\{1,\ldots,n\}$
- Kjent reisekostnad mellom byer c_{ij}
- Hver by har åpningstid og betjeningstid

$$[e_i, l_i] \qquad s_i$$

- Handelsreisende skal foreta rundtur
- Løsning kan representeres ved permutasjon
- S er mengden av alle (lovlige) permutasjoner $\{\pi_p\}_{p=1}^{(n-1)!}$
- f er summen av reisekostnader

$$f(\pi) = \sum_{i=1}^{n-1} c_{\pi(i),\pi(i+1)} + c_{\pi(n),\pi(1)}$$

Alternativ DOP representasjon: Matematisk program

$$\min \quad f(\vec{x}) \quad \text{slik at}$$

$$g_i(\vec{x}) = 0,\ i = 1, \ldots, m$$

$$h_j(\vec{x}) \leq 0,\ j = 1, \ldots, n$$

$\vec{x}$ vektor av diskrete beslutningsvariable

Hva er S?

Alternativ DOP-representasjon
Constrained Optimization Problem (COP)

- Sentrale problemformuleringer i AI
- Constraint Satisfaction Problem (CSP)
- Constrained Optimization Problem (COP)
- Alternativ til Matematisk Programmering
- Eksempel: Kryptoaritmetikk, dronningoppgaven, ...

$$\begin{array}{r} \text{S E N D} \\ + \quad \text{M O R E} \\ = \text{M O N E Y} \end{array}$$

$$D, E, M, N, O, R, S, Y \in \{0, \ldots, 9\}$$

$$(D + E) \bmod 10 = Y$$

$$\vdots$$

Alternativ DOP-representasjon
Constrained Optimization Problem (COP)

Gitt $\vec{x} = (x_1, \ldots, x_n)$

$D = (D_1, \ldots, D_n) = \left(\{v_{11}, \ldots v_{1k_1}\}, \ldots, \{v_{n1}, \ldots v_{nk_n}\} \right)$

$f : D_1 \times \cdots \times D_n \rightarrow \Re$

$C^{(k)} = \{c_1^{(k)}, \ldots, c_{i_k}^{(k)}\}, \quad k = 1, \ldots n$

$c_j^{(k)} \subseteq D_{\pi_j^{(k)}(1)} \times \cdots \times D_{\pi_j^{(k)}(k)}$

min $f(\vec{x} \perp \vec{v})$ slik at

$\vec{v} \in c_j^{(k)}, j = 1, \ldots i_k, \forall k$

DOP Eksempel: Tilordningsproblemet

- n personer
- n oppgaver
- det koster $c_{i,j}$ å la person i utføre oppgave j
- finn minimal kost tilordning:

$$x_{i,j} = \begin{cases} 1 & \text{hvis person i utfører oppgave j} \\ 0 & \text{ellers} \end{cases}$$

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j}\, x_{i,j} \quad \text{s.a.}$$

$$\sum_{j=1}^{n} x_{i,j} = 1 \quad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} x_{i,j} = 1 \quad j = 1, \ldots, n$$

- hvordan gis en probleminstans?
- hvordan kan løsning representeres?
- hva er mengden **S**?
- dårlig case for lokalsøk ...

DOP Eksempel: TSPTW

$$x_{i,j} = \begin{cases} 1 & \text{hvis by j følger rett etter by i} \\ 0 & \text{ellers} \end{cases}$$

a_i ankomsttid node i

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j}\, x_{i,j} \quad \text{s.a.}$$

$$\sum_{j=1}^{n} x_{i,j} = 1 \quad i = 1,\ldots,n$$

$$\sum_{i=1}^{n} x_{i,j} = 1 \quad j = 1,\ldots,n$$

$$e_i \le a_i \le l_i \quad i = 1,\ldots,n$$

$$a_j \ge x_{i,j}\left(a_i + s_i + c_{i,j}\right)$$

- Hvordan gis en probleminstans?
- Hvordan kan løsning representeres?
- Hva er mengden **S**?
- Er modellen riktig?

DOP Eksempel: Ryggsekkproblemet

- n ”artikler” {1,...,n} tilgjengelig, vekt c_i verdi v_i
- et utvalg skal pakkes i ryggsekk med kapasitet C
- bestem utvalg av artikler som maksimerer verdi

$$x_i = \begin{cases} 1 & \text{hvis artikkel i er med i ryggsekken} \\ 0 & \text{ellers} \end{cases}$$

$$\max \sum_{i=1}^{n} v_i x_i \quad \text{s.a.}$$

$$\sum_{j=1}^{n} c_i x_i \leq C$$

- hvordan gis en probleminstans?
- hvordan kan løsning representeres?
- hva er mengden **S**?

Hvordan finne løsninger?

- Eksakte metoder
  - Systematisk generering (Eksplisitt enumrering, generer og test)
  - Implisitt enumrering
    - dele opp i enklere problemer
    - løse enklere problem eksakt
- Triviell løsning
- Inspeksjon av probleminstansen
- Konstruktiv, metode
  - gradvis oppbygging ved grådig heuristikk
- Løse enklere problem
  - fjerne/modifisere føringer
  - modifisere objektfunksjon

Eksempel: TSP

Tidligere løsning:

1 2 7 3 4 5 6 1 (184)

Triviell løsning:

1 2 3 4 5 6 7 1 (288)

Grådig konstruksjon:

1 3 5 7 6 4 2 1 (160)

	1	2	3	4	5	6	7
1	0	18	**17**	23	23	23	23
2	**2**	0	88	23	8	17	32
3	17	33	0	23	**7**	43	23
4	33	**73**	4	0	9	23	19
5	9	65	6	65	0	54	**23**
6	25	99	2	**15**	23	0	13
7	83	40	23	43	77	**23**	0

Eksempel: Ryggsekkproblem

	1	2	3	4	5	6	7	8	9	10
Verdi	79	32	47	18	26	85	33	40	45	59
Størrelse	85	26	48	21	22	95	43	45	55	52

- Ryggsekk med kapasitet 101
- 10 ”artikler” (prosjekter, ...) 1,...,10
- Triviell løsning: tom ryggsekk, verdi 0
- Grådig løsning, prøv artiklene etter verdi:
  - (0000010000), verdi 85
  - bedre forslag til grådig heuristikk?

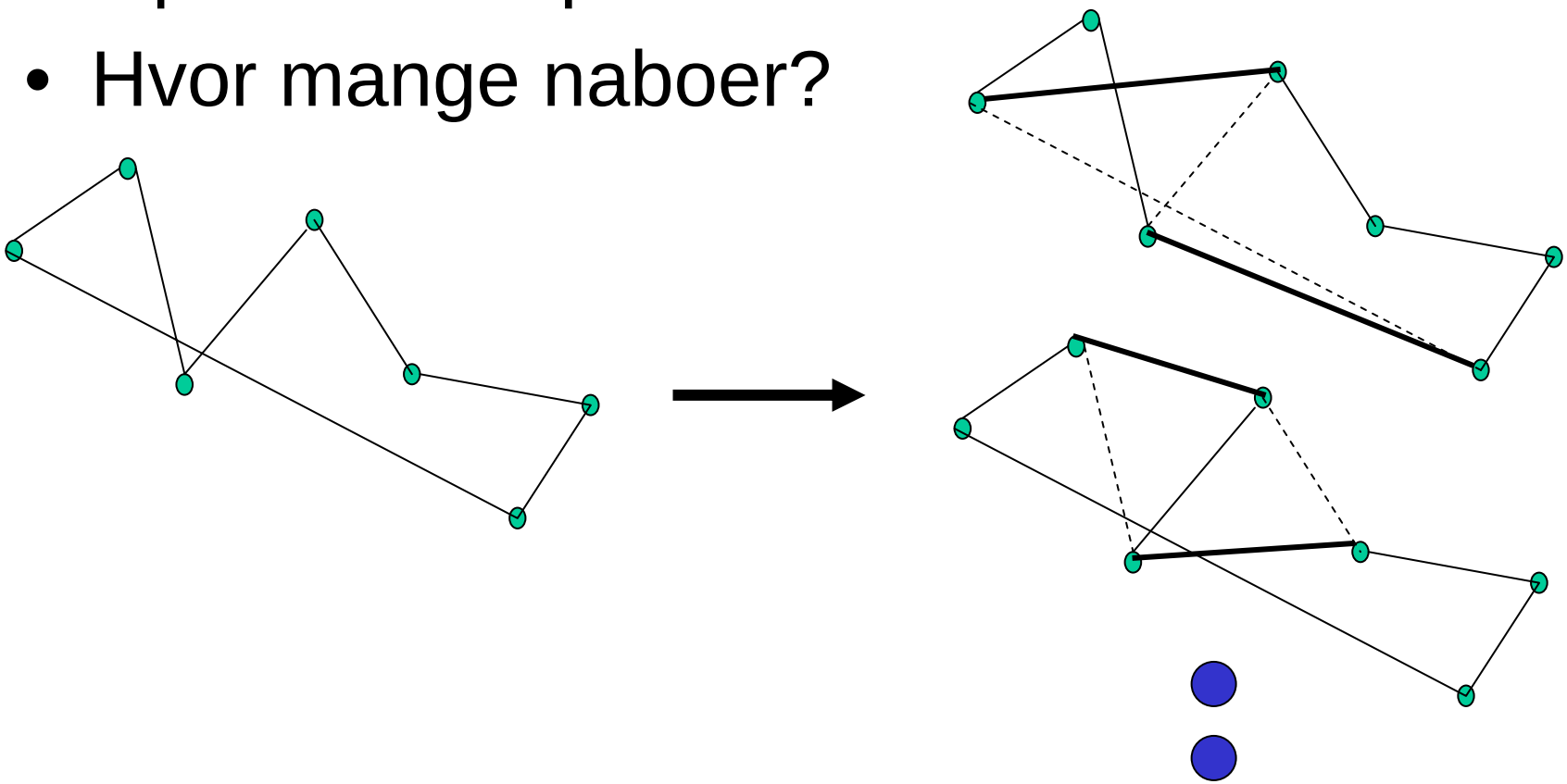
	1	2	3	4	5	6	7	8	9	10
Verdi/størrelse	0.93	**1.23**	0.98	0.86	**1.18**	0.89	0.77	0.89	0.82	**1.13**

Gitt en løsning, hvordan finne bedre løsning?

- Modifikasjon av gitt løsning gir ”naboløsning”
- En viss type **operasjon** på løsningen gir et sett med naboer, et **nabolag**
- Evaluering av naboer
  - objektfunksjon
  - tillatt?

Eksempel: TSP

- Operator: 2-opt
- Hvor mange naboer?



Eksempel: Ryggsekk-instans

	1	2	3	4	5	6	7	8	9	10
Verdi	79	32	47	18	26	85	33	40	45	59
Størrelse	85	26	48	21	22	95	43	45	55	52
	0	0	1	0	1	0	0	0	0	0

- Anta vi ser på løsning 0010100000 verdi 73
- Naturlig operator: ”Flip” en bit, dvs.
  - hvis artikkelen er i ryggsekken, ta den ut
  - hvis artikkelen ikke er i ryggsekken, ta den med
- Noen naboer:
  - 0110100000 verdi 105
  - 1010100000 verdi 152, ikke tillatt
  - 0010000000 verdi 47

Definisjon: Nabolag

La (S,f) være en DOP-instans. En nabolagsfunksjon er en avbildning

$N : S \rightarrow 2^S$

som for gitt løsning $s \in S$ definerer

et nabolag av løsninger $N(s) \subseteq S$

som på et vis er ”i nærheten” av s

$t \in N(s)$ sies å være nabo til s

Nabolagsoperator

- Oftest defineres nabolagene ved en gitt type operasjon på løsningen
- Oftest enkle operasjoner
  - fjerning av element
  - tillegg av element
  - bytte av to eller flere elementer i løsning
- Flere nabolag - kvalifiseres med operator

$$N_\sigma(s), \sigma \in \Sigma$$

Lokalsøk (nabolagssøk)

- Utgangspunkt i initiell løsning
- Iterativt søk i nabolag etter bedre løsning
- Sekvens av løsninger $s_{k+1} \in N_\sigma(s_k), k = 0, \ldots$
- Strategi for hvilken løsning i nabolaget som aksepteres som neste løsning
- Stoppkriterier

- Hva skjer når nabolaget ikke inneholder bedre løsning?

Definisjon: Lokalt optimum

La (S,f) være en DOP-instans og la N være en nabolagsfunksjon. En løsning $\hat{s}$ er **lokalt optimal** (minimal) med hensyn på N dersom:

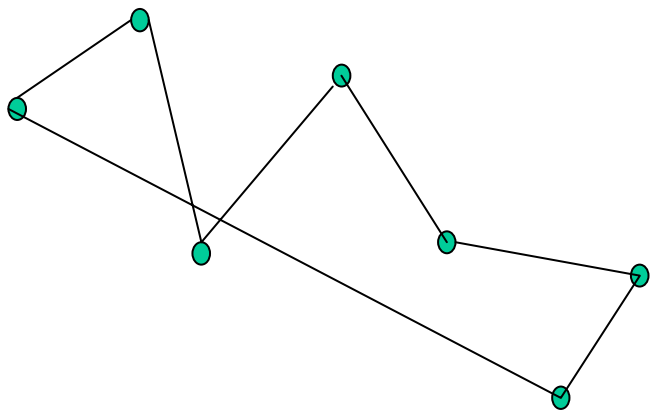
$$f(\hat{s}) \leq f(t), \forall t \in N(\hat{s})$$

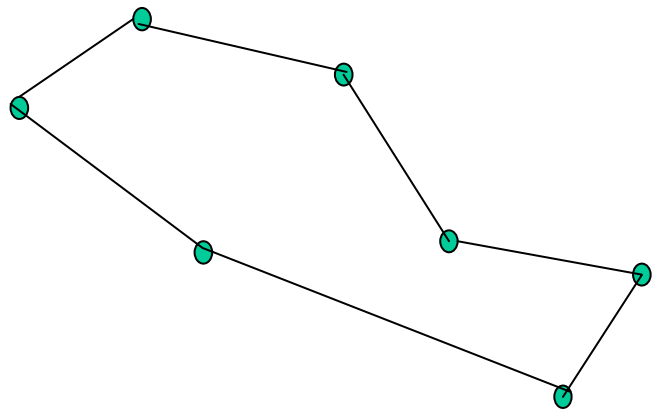
Vi betegner mengden av lokalt optimale løsninger med $\hat{S}$

NB! Lokal optimalitet er relativt til nabolag

Eksempel: TSP

- Operator/Nabolag: 2-opt
- Lokalt optimal (2-optimal) løsning





Definisjon: Eksakt nabolag

La (S,f) være en DOP-instans og la
N være en nabolagsfunksjon. N er **eksakt** dersom:

$$\hat{S} \subseteq S^*$$

Med flere ord:
N er eksakt dersom lokale optima for nabolaget N også er globale optima.

Lokalsøk (nabolagssøk)

- Alternative søkestrategier
- Aksepter første forbedrende løsning (”First Accept”)
- Søk hele nabolag
  - gå til beste **forbedrende** løsning (”Steepest Descent”, ”Hill Climbing” ”Iterative Improvement”)
  - alltid gå til beste løsning i nabolag (”Best Neighbor”)
- Andre strategier?

Local_Search (S,f,N,strategy)

```
*/ strategy is ”First Accept” or ”Best Accept”
current:=Init_Solution(S,f)
incumbent:=current            */ best solution until now
local_optimum:=false
while not local_optimum do
    (current,incumbent,local_optimum):=
       Search_the_Neighborhood
       (current,N(current),f,strategy,incumbent)
    if local_optimum return incumbent
od
```

Search_the_Neighborhood (current,Neighbors,f,strategy,incumbent)

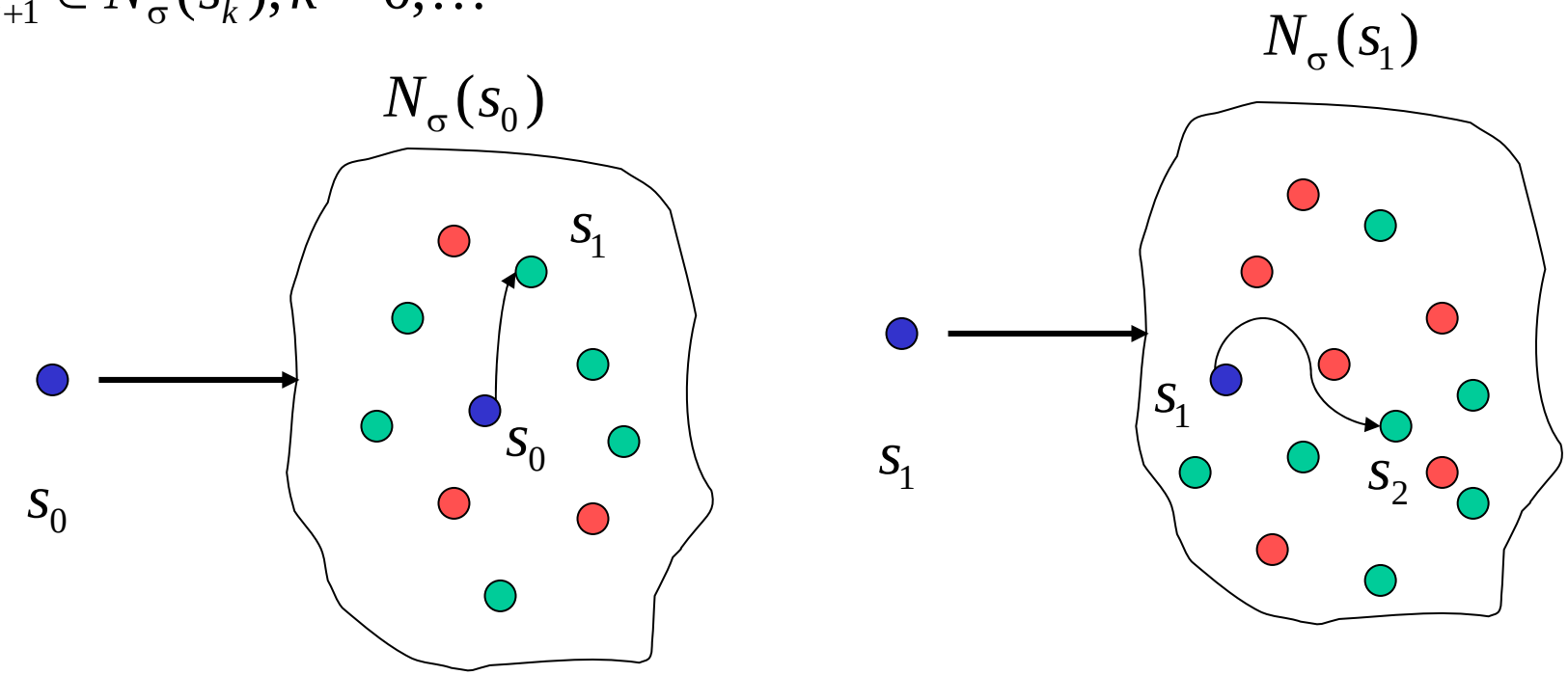
```
best_neighbor:=current
for t in Neighbors do
     if f(t) < f(best_neighbor) then best_neighbor:=t
      if f(t) < f(incumbent) then
          if strategy=”First Accept” then
            return (t,t,false) else
            incumbent:=t                */ strategy is ”Best Accept”
          fi
     fi
od
return (best_neighbor,incumbent,best_neighbor=current)
```

Observasjoner

- ”First Accept” og ”Steepest Descent” stopper i lokale optima
- Dersom nabolaget N er eksakt, er lokalsøk med disse strategiene (eksakte) optimeringsalgoritmer
- Lokalsøk kan betraktes som traversering i en rettet graf (”nabolagsgrafen”), der nodene er medlemmene i S og N definerer topologien (nodene merket med kostnad) og f definerer ”topografien”

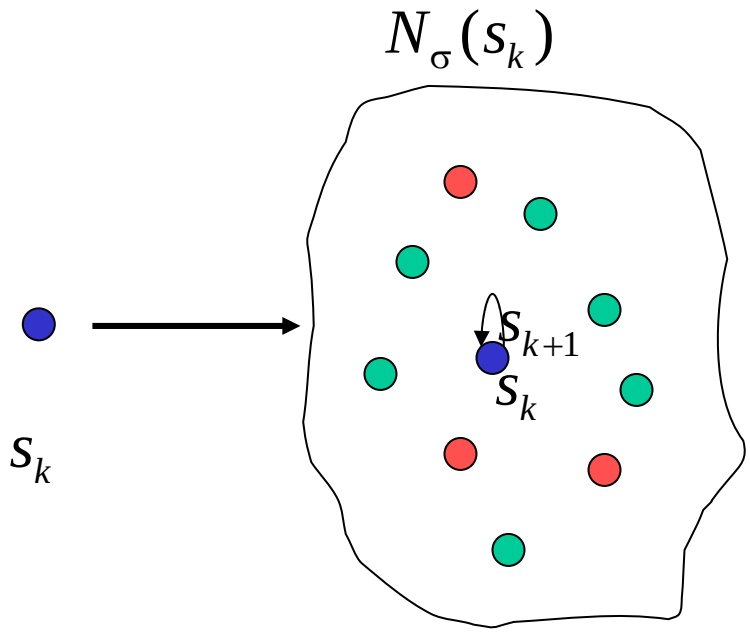
Lokalsøk: Traversering av nabolagsgrafen

$s_{k+1} \in N_\sigma(s_k), k = 0, \ldots$

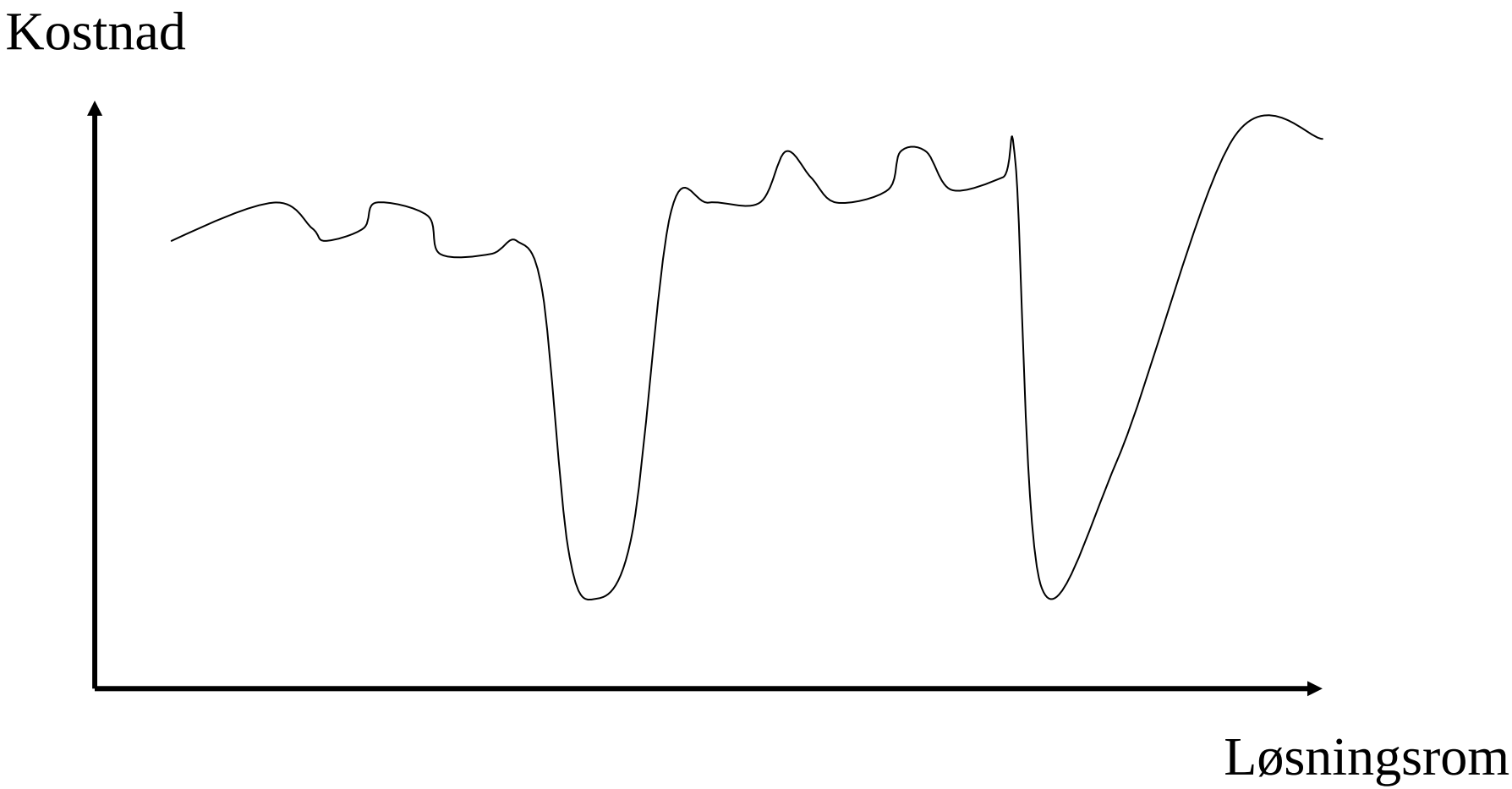


Et *flytt* er prosessen å velge en gitt løsning i nabolaget til nåværende løsning som nåværende løsning for neste iterasjon

Lokalsøk: Lokalt optimum



Lokale og globale optima



Eksempel på lokalsøk

- Simpleksalgoritmen for Lineærprogrammering (LP)
- Simpleks Fase I gir initiell (brukbar) løsning
- Fase II gir iterativ forbedring mot optimal løsning (hvis slik finnes)
- Nabolaget defineres av simpleks-polytopen
- Strategien er ”Iterative Improvement”
- Flyttene bestemmes av pivoteringsregler
- Nabolaget er eksakt, dvs.
  Simpleks er en optimeringsalgoritme (for visse pivoteringsregler)

Lokalsøk

- Gammel idé, utvikling siste par tiår
- Populær metode for praktisk problemløsning av harde problemer
- Generell metode, tilpasningsvennlig
- ”Anytime”-metode, kan avbrytes ”når som helst” etter at initiell løsning er funnet
- Ofte effektivt, god løsning etter kort tid
- Effektivitet avhengig av initiell løsning og nabolag
- Nabolag bør velges ut fra problemstruktur
- Eksakte metoder er å foretrekke dersom de er effektive nok

Lokalsøk

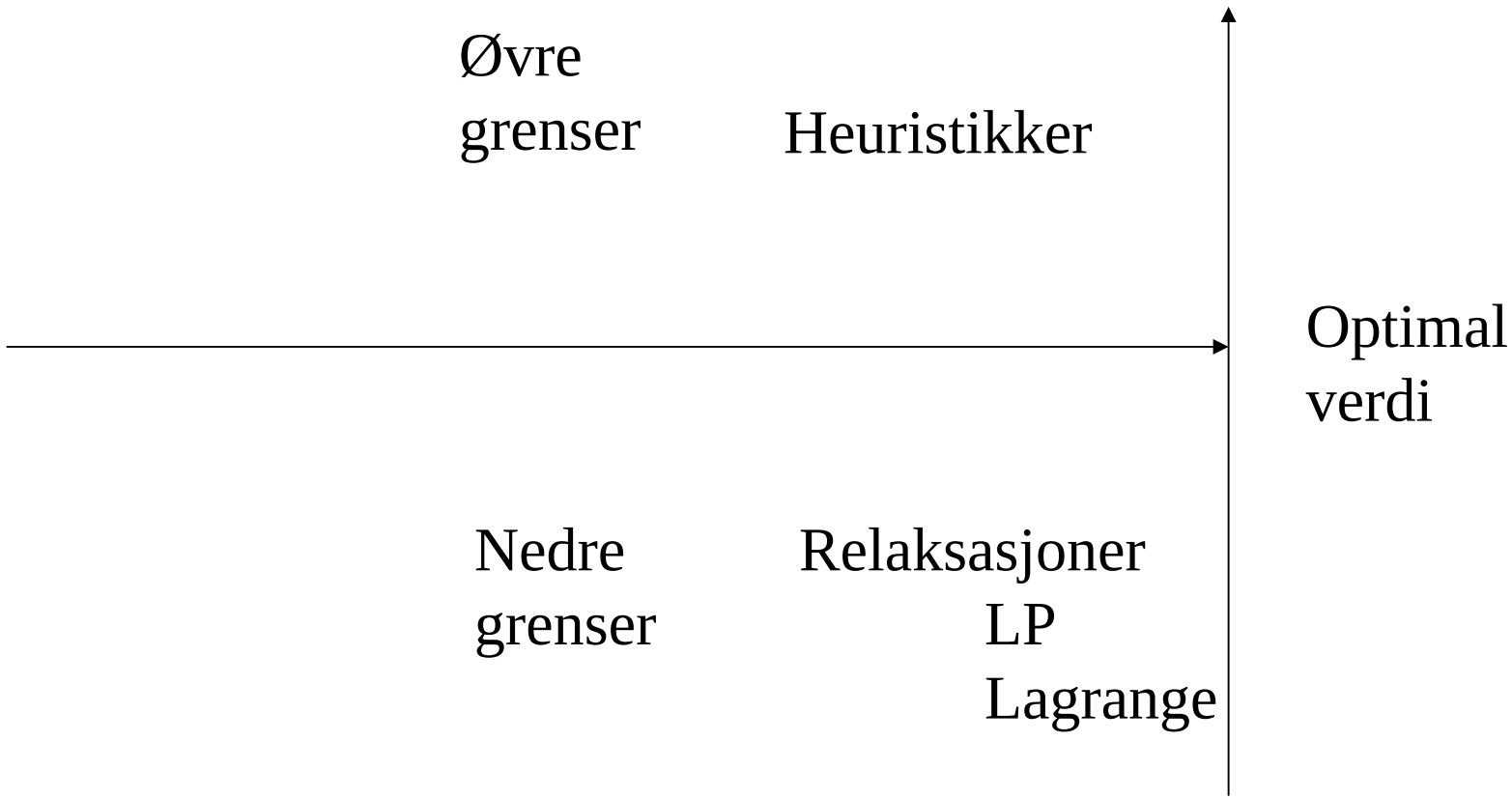
- Hovedutfordring:
  - finne gode nabolag
  - størrelse vs. kvalitet
- Andre utfordringer
  - initiell(e) løsning(er)
  - effektiv evaluering av flytt (inkrementell evaluering)
  - håndtering av føringer
  - søkestrategi
  - stoppkriterier
  - ytelse
- Ytelse ofte bedre enn enkle, grådige heuristikker og heuristikker med ytelsesgaranti
- Hvordan evaluerer vi ytelsen?

Lokalsøk - Ulemper

- Lokalt optimum kan være langt fra globalt optimum
- ”Grådig” metode (Iterative improvement)
- ”Blind” metode, ingen bruk av informasjon under søk
- Ofte sterkt avhengig av initiell løsning og nabolag
- Manglende ytelsesgaranti

Evaluering av heuristikker

- øvre og nedre grenser (ved minimering)



Hovedutfordring i lokalsøk

Hva skal vi gjøre for å unngå
at lokalsøk stopper
i lokalt optimum?

Leksjon 2 - Oppsummering

- Eksempler på DOP
- Alternative formuleringer
- Definisjon nabolag, -operator
- Lokalsøk
- Definisjon lokalt optimum
- Eksakt nabolag
- Prosedyre for lokalsøk
- Traversering av nabolagsgraf
- Kommentarer, ulemper, hovedutfordring

Leksjon 3 - Oversikt

- Tilfeldig søk
- Simulert størkning
- Terskelakseptanse

INF-MAT-5380

http://www.uio.no/studier/emner/matnat/ifi/INF-MAT5380/

Leksjon 2