TAOP61 Optimering av realistiska sammansatta system

- Examinator:
  - Kaj Holmberg
  - kaj.holmberg@liu.se, 013-282867
  - Kurshemsida: http://courses.mai.liu.se/GU/TAOP61/
- Litteratur:
  - Kaj Holmberg: Optimering (Liber, 2018)
  - Kaj Holmberg: Introduktion till matematiska dekompositionsmetoder för strukturerade optimeringsmodeller (2019), Bokakademin
  - Kaj Holmberg: Fallbeskrivningar + projektinfo
- Kursdelar:
  - Föreläsningar 6 st (12 h)
  - Lektioner/lab 4 st (8 h)
  - Projekt 4 st (3 eller 4-personsgrupper) (24 h + ?)
  - Presentationer
- Examination:
  - Skriftliga projektredovisningar
  - Lisam Test
  - Muntliga projektredovisningar (en per grupp) + opposition
  - Skriftlig tenta (betyg)

Speciellt med denna kurs

- Mycket verklighetsnära projekt
- Avancerade optimeringsmetoder
- Ni ska lösa stora svåra problem
- Projektarbetet i princip självständigt
- Ni arbetar och redovisar (och tar ansvar) i grupp
- Undervisningen ger grunder för projekten
- Undervisningen är inte heltäckande
- Projekten innehåller:
  - Modellering
  - Programmering (Matlab/Python)
  - Användande av avancerade program
  - Felsökning
  - Avrapportering
  - Tävling
- Värsta matematiken: Dekompositionsmetoder

Uppdateringar

- 2021: Lisam Test för viss rapportering av projekten.
- 2020: Distansundervinsning: Filmer.
- 2019: Kortat kompendium om dekompositionsmetoder (lite mer i nya boken).
- 2019: Sparbeting, lite mindre undervisning om dekompositionsmetoder.
- 2018: Annan utformning av tentan. (Mindre snack.)
- 2018: Opposition på presentationer.
- 2018: Presentationer anpassade till antal studenter.

Kursplan

Fö 1: Introduktion, formulering av strukturerade modeller, dekomposition som ide.
Introduktion till projekt 1.
Le 1: Modellformulering.
Proj 1-3: Projekt 1 (Returpack).

Fö 2: Dynamisk programmering.
Introduktion till projekt 2.
Le 2: Dynamisk programmering.
Proj 4-6: Projekt 2 (laddhybrid).

Kursplan

Fö 3-5: Lagrangedualitet, dekompositionsmetoder.
Introduktion till projekt 3.
Le/la 3-4: Dekompositionsmetoder.
Proj 7-9: Projekt 3 (elnät).

Fö 6: Brevbärarproblem, heuristiker, relevanta omgivningar för lokalsökning, ruttplanering.
Introduktion till projekt 4.
Le 5 (själva): Brevbärarproblem, heuristiker.
Proj 10-12: Projekt 4 (snöröjning).

Sem 1-2: Presentationer av projekt.

Repetition

- Ni:
  - Grundkursen!
  - Modellering av optimeringsproblem.
  - Matematisk notation: matriser, vektorer, index, summor.
  - Grafisk lösning.
  - KKT-villkoren.
  - LP-dual.
  - Brevbärarproblem.
  - Heuristiker.
- Jag:
  - Modellering av optimeringsproblem.
  - (Matematisk notation: matriser, vektorer, index, summor.)
  - (Brevbärarproblem.)
  - Heuristiker.

Optimering

- Matematisk bas.
- Löser verkliga problem.
- Använder speciellt utvalda metoder.
- Specifika svårigheter:
  - Konstruera relevant modell.
    - ⋆ Ta med relevanta saker (t.ex. kostnader, fysik, miljö).
  - Lös modellen.
    - ⋆ Välj/använd lämplig metod.
  - Kontrollera lösningen.
    - ⋆ Konstigheter i lösningen kan tyda på brister i modell och/eller data.
  - Finn och korrigera fel.
    - ⋆ En komplicerad modell blir sällan korrekt på första försöket.

Långsiktiga mål med kursen

- Känna igen komplexa optimeringsproblem.
- Kunna formulera krångliga problem matematiskt.
- Förstå principerna bakom vissa avancerade metoder.
- Kunna välja lämplig lösningsmetod.
- Kunna använda tillgänglig programvara.
- Medverka vid utveckling av ny programvara.
- (Räkna för hand.)

Litet (fånigt) exempel:

Gör en plåtlåda med maximal volym,
då total plåtyta inte är större 100 cm^2 och höjden inte överstiger 4 cm.

Variabler (det vi ska bestämma):

x_1 : höjd (i cm)

x_2 : bredd (i cm)

x_3 : djup (i cm)

Målfunktion:

Max $V = x_1x_2x_3$ (maximera volymen)

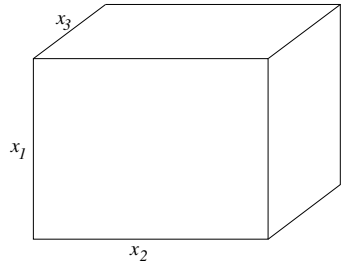
Bivillkor:

$2x_1x_2 + 2x_1x_3 + 2x_2x_3 \leq 100$ (begränsad yta)

$x_1 \leq 4$ (begränsad höjd)

$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$ (fysikaliska begränsningar, negativa mått finns ej)

Lådan



Optimallösning:

$x_1 = 4$ cm (höjd)

$x_2 = 4.124$ cm (bredd)

$x_3 = 4.124$ cm (djup)

Volym: 68.031 cm^3.

Litet dumt exempel:

Misslyckad modellering:

Glömt bivillkoret på begränsad area.
Följd: Optimum obegränsat, dvs. oändligt stor låda.

Glömt kraven $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$.
Följd: Optimum obegränsat, låt t.ex. x_1 och x_2 bli negativa.
Då blir volymen positiv och stor, medan arean blir negativ.

I båda fallen ser vi att lösningen är dum.
Men den är optimal.

Hur kan man misslyckas?

1. Lös problemet felaktigt, dvs. finn en lösning som inte är optimal, utan sämre och/eller otillåten.
   Dålig metod.
2. Lös fel problem, dvs. finn korrekt optimallösning till fel modell.
   Dålig modell.
3. Både 1 och 2, dvs. finn fel lösning till fel modell.
   Dålig modell och metod.

2 är vanligast.

2 och 3 är svåra att upptäcka/korrigera.

Hur man använder optimering: Operationanalys

1. **Formulering av problemet.** Finns det ett problem? Vad vill man optimera? Vilka begränsningar finns?
2. **Konstruktion av en matematisk modell.** Definiera variabler, målfunktion samt bivillkor. Är resultatet en LP-, ILP eller HP-modell?
3. **Insamling av data.**
4. **Lösning av det matematiska problemet.** Välj lämplig optimeringsmetod.
5. **Utvärdering av resultat (och modell).** Är resultatet realistiskt, lämpligt, vettigt, "bra"? Om inte, gå till 2.
6. **Använd resultatet.**

Mål när man gör modellen

- Få med allt relevant, dvs. som påverkar vilken lösning som är optimal.
- Undvik det som är irrelevant, dvs. som inte påverkar vilken lösning som är optimal.
- Modellen ska vara korrekt, dvs. göra det man vill att den ska göra.
- Modellen ska vara lösbar, dvs. gå att lösa på rimlig/tillgänglig tid.
- Data (koefficienter) ska kunna tas fram.
- De förenklingar man kan tvingas göra ska vara medvetna och genomtänkta.
- Undvik onödiga komplikationer, såsom olinjäriteter.
- Välj målfunktion.

Modellering

Modellering av komplexa sammansatta situationer

(nästan alla realistiska problem är sådana)

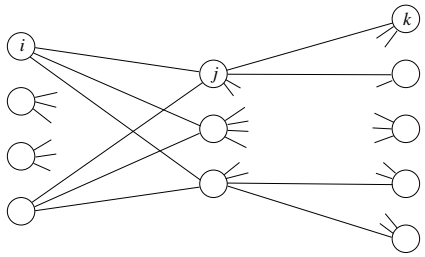
kräver flera olika sorters variabler

och flera olika grupper av bivillkor.

Modellering: Exempel

Vi ska skicka varor från fabriker, i, via lager, j, till affärer, k.

20 fabriker, 10 lager, 40 affärer.



Möjlig variabeldefinition:

x_{ijk} = antal enheter som skickas från fabrik i via lager j till affär k (8000)

Bättre:

z_{ij} = antal enheter som skickas från fabrik i till lager j (200)

y_{jk} = antal enheter som skickas från lager j till affär k (400)

Modellering: Exempel

Fabrik i har S_i enheter tillgängliga,
affär k vill ha D_k enheter, och
lager j kan hantera maximalt L_j enheter.

Med x:

$$\sum_j \sum_k x_{ijk} \leq S_i \quad \text{för alla } i$$
$$\sum_i \sum_j x_{ijk} = D_k \quad \text{för alla } k$$
$$\sum_i \sum_k x_{ijk} \leq L_j \quad \text{för alla } j$$

20+40+10=70 bivillkor

Bivillkorsmatrisens storlek blir 70 x 8000.

Modellering: Exempel

Fabrik i har S_i enheter tillgängliga,
affär k vill ha D_k enheter, och
lager j kan hantera maximalt L_j enheter.

Med z och y:

$$\sum_j z_{ij} \leq S_i \quad \text{för alla } i$$
$$\sum_j y_{jk} = D_k \quad \text{för alla } k$$
$$\sum_i z_{ij} = \sum_k y_{jk} \quad \text{för alla } j$$
$$\sum_i z_{ij} \leq L_j \quad \text{för alla } j$$

20+40+10+10=80 bivillkor

Bivillkorsmatrisens storlek blir 80 x 600.

Modellering: Exempel

Omvandling: Varorna görs om i lagren, så att en enhet in bara blir 0.7 enheter ut.

(Ibland är flödet på ena sidan en annan sort än flödet på andra.)

Ändring av bivillkor:

$$\sum_i z_{ij} = \sum_k y_{jk} \quad \text{för alla } j$$

ersätts av

$$\sum_i 0.7 z_{ij} = \sum_k y_{jk} \quad \text{för alla } j$$

Mer generellt:

$$\sum_i a_j z_{ij} = \sum_k y_{jk} \quad \text{för alla } j$$

Modellering: Exempel

Flera tidsperioder:

z_{ij}^t = antal enheter som skickas från fabrik i till lager j under tidsperiod t

y_{jk}^t = antal enheter som skickas från lager j till affär k under tidsperiod t

Tillgång och efterfrågan kan variera:

$$\sum_j z_{ij}^t \leq S_i^t \quad \text{för alla } i \text{ och } t$$
$$\sum_j y_{jk}^t = D_k^t \quad \text{för alla } k \text{ och } t$$
$$\sum_i z_{ij}^t = \sum_k y_{jk}^t \quad \text{för alla } j \text{ och } t$$
$$\sum_i z_{ij}^t \leq L_j \quad \text{för alla } j \text{ och } t$$

Notera: 100 tidsperioder ger 100 gånger flera variabler. (60 000)

Modellering: Exempel

Men problemet kan lösas separat för varje tidsperiod
om ingen binder samman tidsperioderna,
såsom lagerhållning:

w_j^t = antal enheter som lagras i lager j mellan tidsperiod t och $t+1$

$$w_j^{t-1} + \sum_i z_{ij}^t = \sum_k y_{jk}^t + w_j^t \quad \text{för alla } j \text{ och } t$$

$$w_j^t \leq K_j \quad \text{för alla } j \text{ och } t \text{ (maxlager)}$$

Dekomposition

Vi kommer senare att gå igenom s.k. dekompositionsmetoder
som är ett angreppssätt för att lösa problem genom att "dekomponera" dem i mindre lättare bitar.

Detta görs genom att relaxera vissa bivillkor
eller genom att fixera vissa variabler.

Därför kan man fundera på vilka bivillkor som skulle kunna relaxeras

eller vilka variabler som skulle kunna fixeras,

så att man får ett mer lättlöst subproblem

för olika strukturerade problem.

Modellering: Lokaliseringsproblemet

Anläggningslokalisering (*facility location*): Vi vill finna bästa platserna för att lokalisera vissa anläggningar, t.ex. fabriker.

I det *diskreta* lokaliseringsproblemet har man ett antal möjliga platser för lokaliseringen,
och man ska välja ut vilken eller vilka av dessa platser som ska användas,
dvs. vilka av de möjliga fabrikerna som ska byggas.

I det *kontinuerliga* lokaliseringsproblemet ska man placera en eller flera anläggningar var som helst i ett område.
I detta problem är koordinaterna för anläggningarna variabler.

Modellering: Lokaliseringsproblemet

Vi har ett antal kunder, med specificerad efterfrågan.

Målet är att tillgodose all efterfrågan till minsta möjliga kostnad.

Målfunktionen innehåller både kostnader för att transportera varorna och för att bygga/installera/köpa anläggningarna.

En viktig aspekt är de *fasta kostnaderna* på anläggningarna.

Vi exemplifierar med fabriker som anläggningar,
men det kan lika gärna handla om att installera många andra saker, såsom telekomutrustning el.dyl.

Modellering: Okapaciterad lokalisering

Vi har m möjliga platser för fabrikerna, och n kunder.

Det kostar f_i att bygga en fabrik på plats i.

Det kostar c_{ij} att transportera kund j:s hela behov, d_j, från plats i.

Vi antar att varje fabrik har mycket stor kapacitet.

Variabeldefinition:

$$y_i = \begin{cases} 1 & \text{om en fabrik byggs på plats } i \\ 0 & \text{om inte} \end{cases}$$

x_{ij} är andelen av kund j:s behov som tas från fabriken på plats i.

Den matematiska modellen ska innehålla en målfunktion med alla kostnader,

samt bivillkor som ser till att kunderna blir nöjda,

och att vi bara använder fabriker som byggs.

Modellering: Okapaciterad lokalisering

Matematisk modell för det okapaciterade lokaliseringsproblemet:

$$\begin{aligned} v^* = \quad \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} f_i y_i & & \\ \text{då} \quad & \sum_{i=1}^{m} x_{ij} = 1 & \forall j & \quad (1) \\ & x_{ij} \le y_i & \forall i,j & \quad (2) \\ & x_{ij} \ge 0 & \forall i,j & \quad (3) \\ & y_i \in \{0,1\} & \forall i & \quad (4) \end{aligned}$$

Bivillkor 1 ser till att kundernas efterfrågan tillgodoses.

Bivillkor 2 förbjuder transporter från platser utan fabriker.

Man skulle kunna lägga till bivillkoret: $\sum_{i=1}^{m} y_i \ge 1$ (5)

som säger att minst en fabrik måste byggas.

Modellering: Okapaciterad lokalisering

Om man fixerar y (till en binär vektor)

dvs. bestämmer var fabriker ska byggas,

och låter I vara index för platser med fabriker (dvs. $I = \{i : y_i = 1\}$),

fås direkt $x_{ij} = 0$ för alla $i \notin I$.

Resten av problemet löses enkelt genom att varje kund får sina varor av den byggda fabriken med lägst transportkostnad:

$x_{\hat{i}j} = 1$ där $\hat{i}$ fås av $\min_{i \in I} c_{ij}$.

Slutsatser: Om y är heltal, blir x automatiskt heltal.

Om y fixeras blir problemet lättlöst.

Modellering: Kapaciterad lokalisering

Om fabriken som byggs på plats i har en begränsad kapacitet, s_i, fås det *kapaciterade* lokaliseringsproblemet.

$$\begin{aligned} v^* = \quad \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} f_i y_i & & \\ \text{då} \quad & \sum_{i=1}^{m} x_{ij} = 1 & \forall j & \quad (1) \\ & \sum_{j=1}^{n} d_j x_{ij} \le s_i y_i & \forall i & \quad (2) \\ & x_{ij} \le y_i & \forall i,j & \quad (3) \\ & x_{ij} \ge 0 & \forall i,j & \quad (4) \\ & y_i \in \{0,1\} & \forall i & \quad (5) \end{aligned}$$

Vi måste lägga till bivillkor 2, som ser till att en fabrik inte skickar mer än den kan producera.

(Bivillkor 3 är inte nödvändigt, men kan hjälpa, se lab 4 i TAOP88.)

Modellering: Kapaciterad lokalisering

Om man fixerar y, fås ett transportproblem (minkostnadsflödesproblem).

Relativt enkelt, men inte lika lätt som utan kapaciteter.

Man skulle kunna lägga till följande bivillkor:

$$\sum_{i=1}^{m} s_i y_i \geq D_{TOT} \qquad (6)$$

där $D_{TOT} = \sum_j d_j$. Bivillkoret säger att den totala kapaciteten hos de byggda fabrikerna måste vara minst lika stor som den totala efterfrågan.

Modellering: Flödesproblem med fasta kostnader

Vanligt minkostnadsflödesproblem, men

för att kunna använda en båge måste man bygga/installera/köpa den,

dvs. betala en fast kostnad.

$$v^* = \min \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{(i,j)\in A} f_{ij}y_{ij} \qquad \text{(FCNF)}$$

$$\text{s.t.} \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b_i \quad \forall i \in N \quad (1)$$

$$x_{ij} \leq d_{ij}y_{ij} \quad \forall (i,j) \in A \quad (2)$$

$$x_{ij} \geq 0 \quad \forall (i,j) \in A \quad (3)$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A \quad (4)$$

Om man fixerar y, fås ett vanligt flödesproblem, men bara med de bågar som har $y_{ij} = 1$.

Modellering: Flervaruflödesproblemet

Ibland består varorna i ett flödesproblem av flera olika sorter.

Vi får då det s.k. *flervaruflödesproblemet*.

Vi har noderna N, bågarna A och varusorterna C.

Kostnaden för att transportera en enhet av varusort k på båge (i,j) är c_{ij}^k (och antas vara linjär).

Nettotillgången av varusort k i nod i betecknas med r_i^k (tillgång positiv, efterfrågan negativ).

Båge (i,j) har kapacitet b_{ij} (gemensamt för alla varusorter).

Variabeldefinition: x_{ij}^k är flödet av varusort k i båge (i,j).

Målet är att minimera kostnaderna då alla nodjämviktvillkor uppfylls.

Modellering: Flervaruflödesproblemet

$$v^* = \min \sum_{k\in C} \sum_{(i,j)\in A} c_{ij}^k x_{ij}^k$$

$$\text{s.t.} \sum_{j:(i,j)\in A} x_{ij}^k - \sum_{j:(j,i)\in A} x_{ji}^k = r_i^k \quad \forall i \in N,\ \forall k \in C \quad (1)$$

$$\sum_{k\in C} x_{ij}^k \leq b_{ij} \quad \forall (i,j) \in A \quad (2)$$

$$x_{ij}^k \geq 0 \quad \forall (i,j) \in A,\ \forall k \in C \quad (3)$$

Bivillkor 1 garanterar nodjämvikt.

Bivillkor 2 ser till att bågkapaciteterna inte överskrids.

LP-problem med $|A||C|$ variabler och $|N||C| + |A|$ bivillkor. (1000 bågar, 100 noder och 100 varusorter ger 100 000 variabler och 11 000 bivillkor.)

Ibland används varusorterna för att särskilja start- och slutnoder.

Då kan antalet varusorter vara i storleksordningen $|N|^2$. (1000 bågar och 100 noder ger då 10 000 varusorter och 10 000 000 variabler.)

Modellering: Komplex design

Det mest generella och komplexa problemet uppstår då man ska designa en hel verksamhet, t.ex. en hel fabrik.

Man kan då ha fasta kostnader på både noder och bågar i ett nätverk.

Dessutom har man flera olika varusorter.

En ytterligare komplikation är att en varusort kan omvandlas till en annan, och att mängden därvid förändras.

Detta gör att vissa koefficienter behövs i omvandlingsvillkoren, och den rena nätverkstrukturen störs ytterligare.

Rimlighetsbedömning

När man har löst ett stort komplicerat problem måste man försöka avgöra om lösningen är korrekt,

dvs. om modellen är korrekt.

Det är ofta lättare att se konstigheter i en lösningen än i en modell.

Hur kan man göra en rimlighetsbedömning av en lösning?

Studera lösningen och vad den innebär.

Lös ett förenklat problem (på annat sätt).

Studera en relaxation av problemet. Ger en optimistisk uppskattning.

Studera en restriktion av problemet. Ger en pessimistisk uppskattning.

Rimlighetsbedömning

Exempel: Lokaliseringsproblemet

- Leveranser från obyggd fabrik.
- Leveranser utöver kapaciteten från byggd fabrik.
- Kund som blir utan leveranser.
- Kund som inte får efterfrågan tillgodosedd.
- Fabrik som byggs men inte gör några leveranser.
- Relaxation: Skicka till varje kund med billigaste transportväg. Öppna billigaste fabrikerna (tills total kapacitet är minst lika med total efterfrågan).
- Restriktion: Finn en tillåten lösning. (Heuristikerna i lab 4 i TAOP88.)

GMPL

GMPL är ett modelleringsspråk.

Man definierar sin modell med mängder och summor

så att det liknar en modell i matematisk form.

Modellen definieras för sig och data för sig.

Flexibelt.

En modellfil kan användas till flera olika datafiler.

GLPK är paketet som innehåller GMPL och lösaren `glpsol`, som körs i en terminal.

AMPL är ett nästan identiskt modelleringsspråk.

Men kommersiellt.

AMPL kan kopplas till olika lösare, bl.a. CPLEX.

GMPL

Dela på modellfil och datafil.

Skriv en modellfil med generella parametrar

som kan köras med olika datafiler.

I GMPL/AMPL måste alla data definieras i modellfilen innan de används i modellen,

och ges numeriska data i datafilen.

Sedan kör man

```
glpsol -m minmodellfil.mod -d mindatafil.dat -o minutfil.txt
```

Det kommer utskrifter på skärmen, bl.a.

```
INTEGER OPTIMAL SOLUTION FOUND
```

eller

```
PROBLEM HAS NO PRIMAL FEASIBLE SOLUTION.
```

Lösningen hamnar på filen `minutfil.txt` (inte på skärmen).

GMPL: Exempel

I modellfilen:

Parametrar:

```
param nbutik;
param ngross;
```

Mängder:

```
set BUTIK := 1..nbutik;
set GROSS := 1..ngross;
```

Data i vektorform:

```
param behov{BUTIK};
```

Data i matrisform:

```
param distBuG{BUTIK,GROSS};
```


GMPL: Exempel

I datafilen:

```
param nbutik := 4;
param ngross := 3;
```

Data i vektorform:

```
param : behov :=
1 100
2 100
3 100
4 100;
```

Data i matrisform:

```
param distBuG : 1 2 3 :=
  1 10 13 22
  2 13 13 26
  3 20 23 29
  4 17 23 12;
```


GMPL: Exempel

I modellfilen:

Variabler:

```
var xBG{BUTIK,GROSS} >=0;
var useG{GROSS} binary;
```

Målfunktion:

```
minimize cost:
sum{i in BUTIK, j in GROSS} tomkost*distBuG[i,j]*xBG[i,j] +
sum{j in GROSS} fkostG[j]*useG[j];
```

Bivillkor:

```
subject to inbutik{i in BUTIK}:
sum{j in GROSS} xBG[i,j] = behov[i];

subject to grosskaptom{j in GROSS}:
sum{i in BUTIK} xBG[i,j] <= gkaptom[j]*useG[j];
```