

# Fast Solver of Closely Related Quadratic Programming Problems

Andreas Halle

June 10, 2013

# PROMAPS

- ▶ Utviklet av Goodtech og MathConsult
- ▶ Kalkulerer leveransepåliteligheten i et nettverk
- ▶ Utfall: Gren i nettverket faller ut
- ▶ Formulert som mange like QP-problemer
- ▶ Flaskehals: QP-løseren
- ▶ Skjermbildet oppdateres hvert 5. minutt
- ▶ [http://www.tu.no/energi/2011/10/07/  
her-beregnes-risikoen-for-svikt-i-kraftnettet](http://www.tu.no/energi/2011/10/07/her-beregnes-risikoen-for-svikt-i-kraftnettet)

# Objektfunksjonen

$$f(x) = x^T \Phi D x + (g - c)^T x$$

- ▶  $x = [x_1 \ x_2 \ \cdots \ x_n]^T$
- ▶  $f(x)$  representerer leveransekostnader ( $E/s$ )
- ▶  $x$  representerer strømmen over grenene ( $W$ )
- ▶  $\Phi$  representerer strømtap ( $1/W$ )
- ▶  $D$  representerer overføringskostnader ( $E/J$ )
- ▶  $g$  representerer kostnader for å generere strøm ( $E/J$ )
- ▶  $c$  representerer leveransepris ( $E/J$ )

# Objektfunksjonen

$$f(x) = x^T H x + b^T x$$

- ▶  $H = \Phi D$  representerer kostnader ( $E/(W^2 s)$ )
- ▶  $b = g - c$  representerer kostnader ( $E/J$ )
- ▶ Minimerer kostnader

# Optimeringsproblemet

Et konvekst QP-problem

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to } Ax = 0, \quad l \leq x \leq u$$

- ▶  $A$  er en  $m \times n$  orientert insidensmatrise for et strømnettverk (rettet graf)
- ▶  $m$  noder,  $n$  grener
- ▶  $l$  er nedre grenkapasitet ( $W$ )
- ▶  $u$  er øvre grenkapasitet ( $W$ )

# Utfall

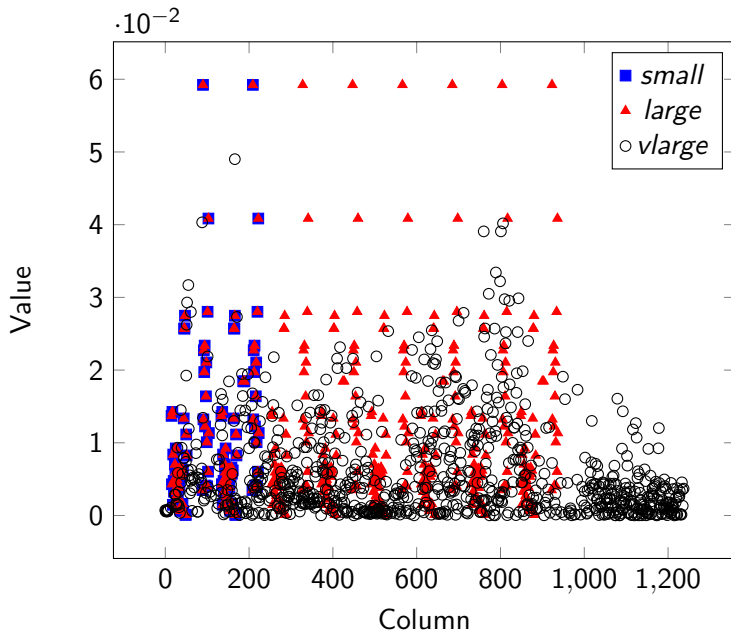
## Modellerer utfall

- ▶ Setter  $l_i = u_i = 0 \Rightarrow x_i = 0$
- ▶ Løse QP-problemet for så mange utfall som mulig
- ▶ Instans er et QP-problem uten utfall
- ▶ Subinstans er en instans med utfall

# Utfall

- ▶  $2^n$  mulige utfall
- ▶ Usannsynlig at det er mange utfall
- ▶ Løse alle subinstanser med maks et gitt antall utfall

# Instanser fra Goodtech





# Instanser fra Goodtech

Table : Størrelse for hver instans

Problemstørrelse	<i>small</i>	<i>large</i>	<i>vlarge</i>
Rader	82	328	1127
Kolonne	238	952	3437
Ikke-nuller A	348	1392	4840
Ikke-nuller H	108	432	894

Table : Verdier i objektfunksjonen

	<i>small</i> og <i>large</i>	<i>vlarge</i>
$\max(h_{ij})$	$2.9614 \times 10^{-2}$	$4.9011 \times 10^{-2}$
$\min(h_{ij})$	$4.9290 \times 10^{-5}$	$1.1026 \times 10^{-5}$
$\text{mean}(h_{ij})$	$5.2864 \times 10^{-3}$	$5.8984 \times 10^{-3}$
$\max(b_i)$	20	20
$\min(b_i)$	-70	-50

# Instanser fra Goodtech

- ▶  $h_{ii} \ll b_i$

$$\mathcal{Q}: \quad \min_{x \in \mathbb{R}^n} \quad x^T H x + b^T x \quad \text{subject to } A x = 0, \quad l \leq x \leq u$$

$$\mathcal{L}: \quad \min_{x \in \mathbb{R}^n} \quad b^T x \quad \text{subject to } A x = 0, \quad l \leq x \leq u$$

- ▶ Optimal løsning til  $\mathcal{Q}$  noteres  $x^*$
- ▶ Optimal løsning til  $\mathcal{L}$  noteres  $\hat{x}$
- ▶  $\Delta$  noterer avviket mellom  $f(x^*)$  og  $f(\hat{x})$

Hvor like er  $\mathcal{L}$  og  $\mathcal{Q}$ ?

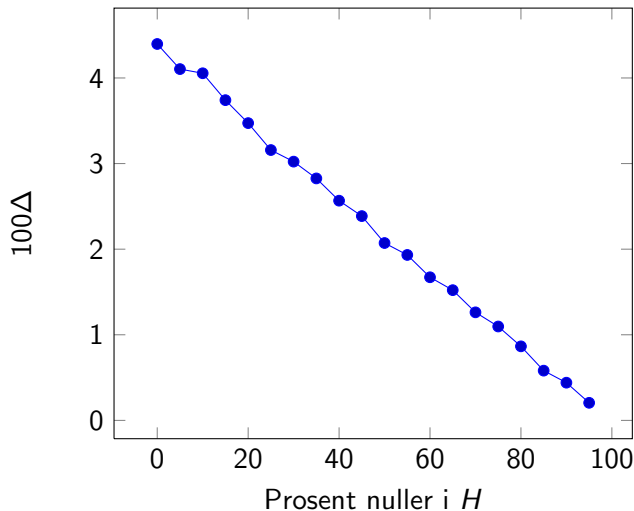


Figure : Avvik som en funksjon av tettheten i  $H$ .

# Hvor like er $\mathcal{L}$ og $\mathcal{Q}$ ?

Oppnår 95% optimal verdi etter løst  $\mathcal{L}$ .

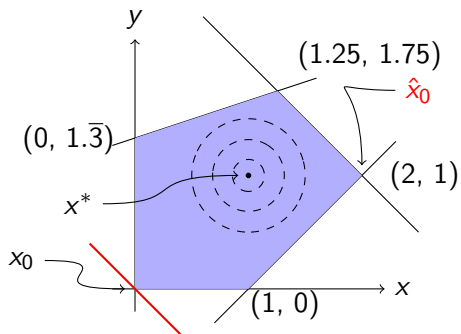
- ▶ Metode basert på successive linear programming (SLP)
- ▶  $x_0 = 0 \Rightarrow$  95% av optimal målfunksjonsverdi etter 1 iterasjon

## Et eksempel

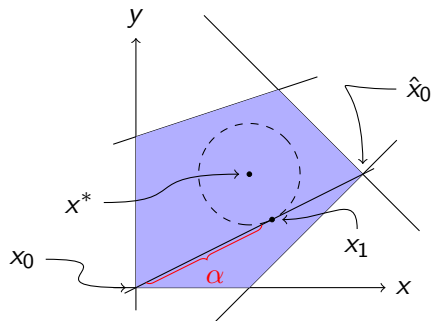
$$\begin{array}{ll} \mathcal{Q} : \text{minimize} & (x-1)^2 + (y-1)^2 - 2 \\ \text{subject to} & x + y \leq 3 \\ & x - y \leq 1 \\ & x + 3y \leq 4 \\ & x, y \geq 0 \end{array}$$

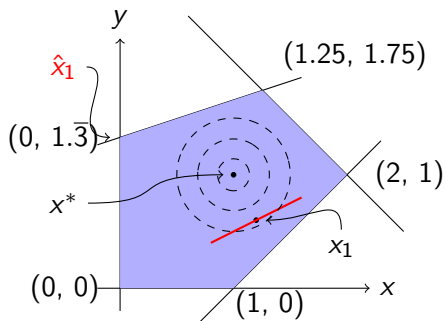
$$\begin{array}{ll} \mathcal{L} : \text{minimize} & -2x - 2y \\ \text{subject to} & x + y \leq 3 \\ & x - y \leq 1 \\ & x + 3y \leq 4 \\ & x, y \geq 0 \end{array}$$

$\mathcal{L}_0$

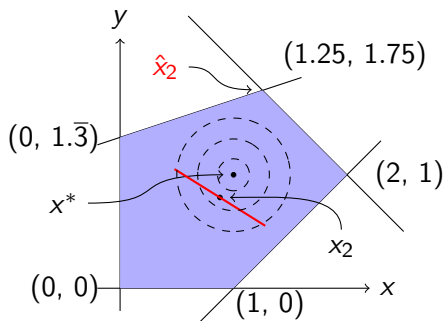


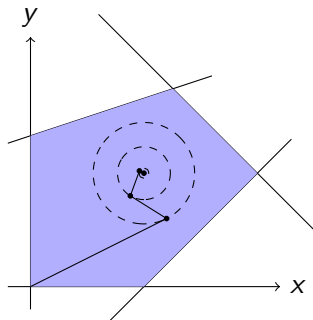
# Linjesøk



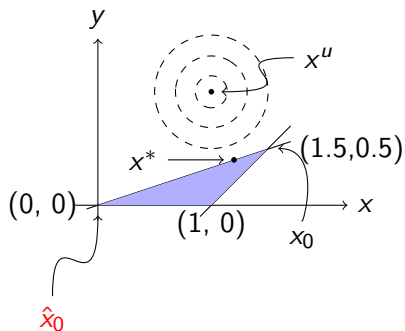
$\mathcal{L}_1$ 



$\mathcal{L}_2$ 



# Endrer et sidekrav



- ▶  $-x + 3y \leq 4$
- ▶  $-x + 3y \leq 0$

# Like optimale løsninger

- Metode for å redusere antall QP-kall

$(3, 0, 7, 0, 4)$

|

|

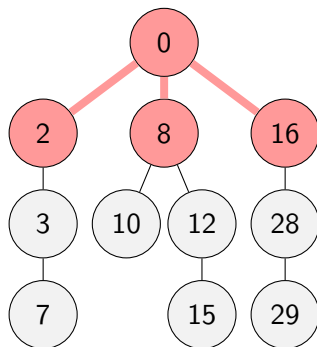
$(3, 0, 7, 0, 4)$

$(3, 0, 7, 0, 4)$

|

$(a, b, 0, c, d)$

# En trekonstruksjon



$$\mathcal{M}_0 = \{\}$$

$$\mathcal{M}_2 = \{2\}$$

$$\mathcal{M}_3 = \{1, 2\}$$

$$\mathcal{M}_7 = \{1, 2, 3\}$$

$$\mathcal{M}_8 = \{4\}$$

$$\mathcal{M}_{10} = \{2, 4\}$$

$$\mathcal{Z}_0 = \{1, 3\}$$

$$\mathcal{Z}_2 = \{2, 3, 5\}$$

$$\mathcal{Z}_3 = \{1, 2, 4, 5\}$$

$$\mathcal{Z}_7 = \{1, 2, 3, 5\}$$

$$\mathcal{Z}_8 = \{1, 4, 5\}$$

$$\mathcal{Z}_{10} = \{2, 3, 4, 5\}$$

$$\mathcal{M}_{12} = \{3, 4\}$$

$$\mathcal{M}_{15} = \{1, 2, 3, 4\}$$

$$\mathcal{M}_{16} = \{5\}$$

$$\mathcal{M}_{28} = \{3, 4, 5\}$$

$$\mathcal{M}_{29} = \{1, 3, 4, 5\}$$

$$\mathcal{Z}_{12} = \{4, 3, 1\}$$

$$\mathcal{Z}_{15} = \{1, 2, 3, 4\}$$

$$\mathcal{Z}_{16} = \{1, 3, 5\}$$

$$\mathcal{Z}_{28} = \{2, 3, 4, 5\}$$

$$\mathcal{Z}_{29} = \{1, 2, 3, 4, 5\}$$

# Eksperiment 1

Vi løser *small* og opp til 2 utfall. 28442 subinstanser

Table : Kjøretid i CPU-sekunder

$\epsilon$	cClp	cSlp	nClp	nSlp
$10^{-1}$	45.51	55.61	72.32	85.51
$10^{-2}$	46.34	55.89	73.11	85.51
$10^{-3}$	51.12	59.04	75.60	85.28
$10^{-4}$	52.46	73.79	77.83	107.39
$10^{-5}$	54.48	232.53	81.16	355.47
$10^{-6}$	65.42	1363.46	93.29	2022.25
$10^{-7}$	70.78	6522.91	100.85	9395.92

# Eksperiment 1

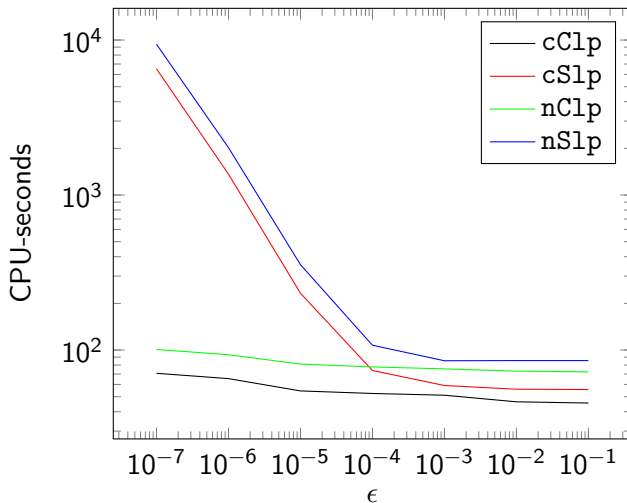


Figure : Kjøretid i CPU-sekunder for å løse *small* og dens subinstanser.

## Eksperiment 2

Table : Kjøretid i CPU-sekunder for å løse de tre instansene.

Implementasjon	<i>small</i>	<i>large</i>	<i>vlarge</i>
cClp	0.52	9.55	76.18
cSlp	0.71	32.88	585.60
nClp	0.65	11.68	157.53
nSlp	0.89	39.87	1173.74



# Tilfeldige instanser

- ▶  $m = \lfloor \frac{7}{20} n \rfloor$
- ▶ 50% null i  $b$  og  $H$

# Eksperiment 3

Table : Kjøretid for å løse tilfeldige instanser med økende  $n$ .  $\beta = 1$ .

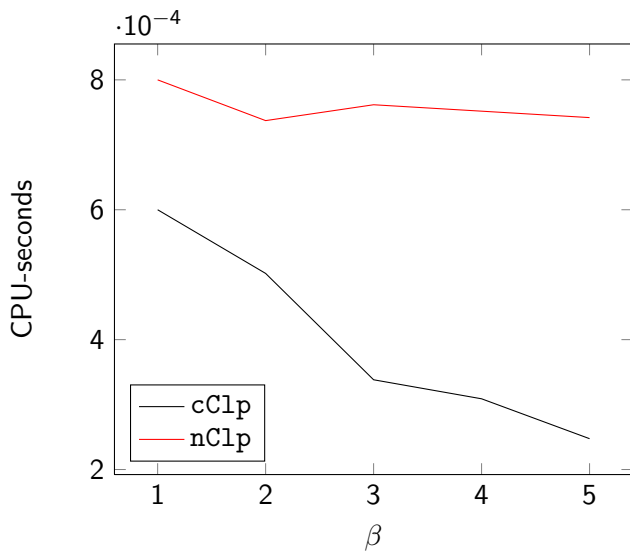
$n$	cClp	nClp	Relativ Speedup
500	4.9	5.9	16.9%
1000	42.1	53.0	20.6%
1500	181.5	234.5	22.6%
2000	547.1	710.2	23.0%

## Eksperiment 4

Table : Kjøretid i CPU-sekunder for  $n = 50$  og økende  $\beta$ .

$\beta$	cClp	nClp	Relativ Speedup	Distinkte løsninger
1	0.03	0.04	25.0%	37.4 (74.3%)
2	0.64	0.94	31.9%	744.3 (58.3%)
3	7.06	15.90	55.6%	9484.7 (45.4%)
4	77.59	188.83	58.9%	82262.5 (32.8%)
5	586.54	1758.23	66.6%	574685.0 (24.2%)

## Eksperiment 4



# Vi konkluderer

- ▶ Raskere jo flere subinstanser vi løser
- ▶ Speedup uavhengig av løser
- ▶ Avslutter med:
- ▶ Løse likt antall subinstanser, raskere: PROMAPS mer responsivt
- ▶ Løse flere subinstanser: Mer nøyaktig analyse