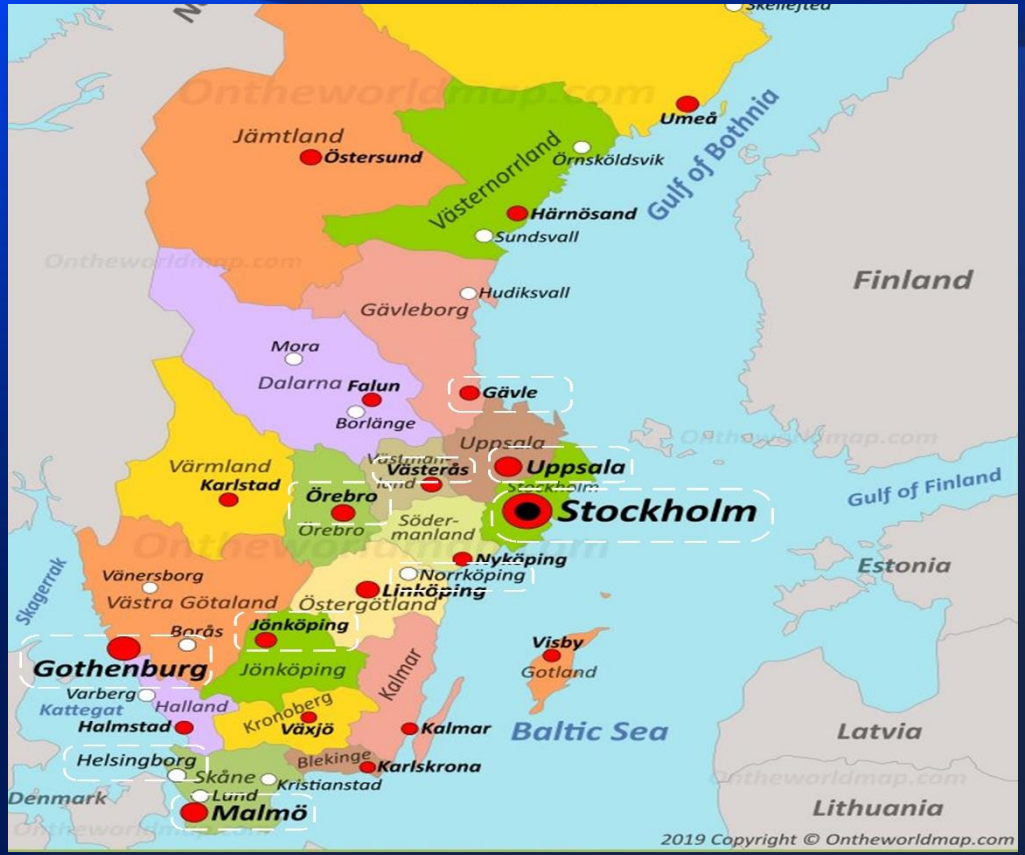
Dispatching Problem in Sweden

By: Niusha Mirhakimi

Operation Research Course

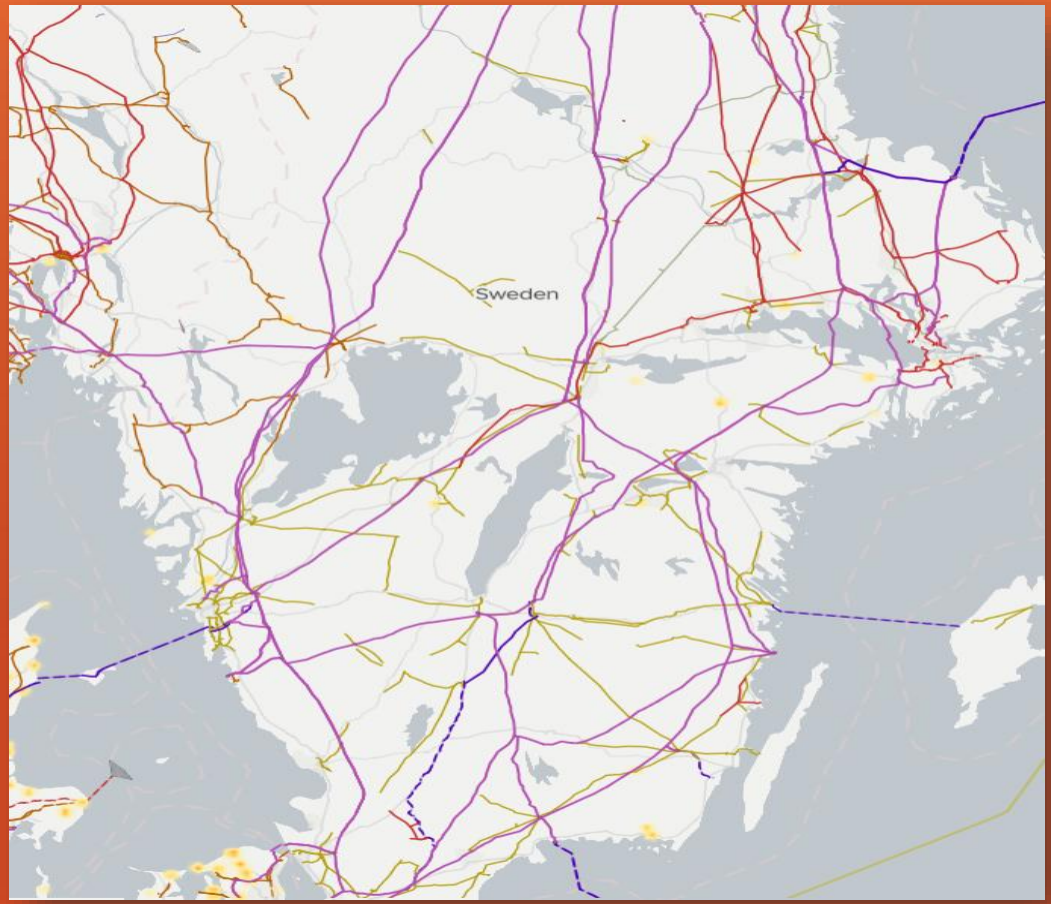
Fall 2020

Cities



- Stockholm
- Västerås
- Uppsala
- Norrköping
- Gothenburg
- Malmö
- Helsingborge
- Jönköping
- Gävle
- Örebro

Transmission Lines Map



Dispatching Problem Graph

Nodes

10 cities:

a. Stockholm
b. Västerås
c. Uppsala
d. Norrköping
e. Gothenburg
f. Malmö
g. Helsingborge
h. Jönköping
i. Gävle
j. Örebro

13 Power Plants:

Forsmark 1
Forsmark 2
Forsmark 3
Oskarshamn 3
Ringhals 1
Ringhals 3
Ringhals 4
Blaiken
Jädraås
Markbygden Wind Farm
Juktan Pumped-Storage
Porjus Hydroelectric
Harsprånget Hydroelectric power station
Harrsele Kraftverk station
Messaure
Letsi
Kilforsen
Trängslet Dam

Power Plants Table

Power Plant	Latitude	Longitude	Capacity(MW)	Type
Forsmark 1	60.405508	18.161366	984	Nuclear
Forsmark 2	60.403966	18.173425	1120	
Forsmark 3	60.402753	18.175163	1167	
Oskarshamn 3	57.416095	16.673137	1400	
Ringhals 1	57.261988	12.111311	865	
Ringhals 3	57.257962	12.106998	1070	
Ringhals 4	57.25665	12.108522	1120	
Blaiken	65.42525	17.332278	250	Wind
Jädraås	60.803278	16.29475	203	
Markbygden Wind Farm	65.416667	20.666667	4000	
Juktan Pumped-Storage Hydroelectric Power Station	64.960185	17.579842	334	Hydroelectric
Porjus Hydroelectric Power Station	66.954281	19.796076	480	
Harsprånget Hydroelectric power station	66.885	19.8148	977	
Harrsele Kraftverk Hydroelectric power station	64.0399	19.5529	223	
Messaure	66.41	20.2	460	
Letsi	66.5045	20.3838	456	
Kilforsen	63.3244	16.4542	415	
Trängslet Dam	61.38	13.73	330	
			Power Sum: 15854	

Cities Table

City	Latitude	Longitude	Population	Demand(MW)
Stockholm	59.334591	18.06324	972,647	2917
Västerås	59.611366	16.545025	127,799	383
Uppsala	59.8498	59.8498	376,354	1129
Norrköping	58.588455	16.188313	137,326	411
Gothenburg	57.70887	11.97456	570,000	1710
Malmö	55.5932	13.0214	316,588	949
Helsingborge	56.0424	12.721	108,334	325
Jönköping	57.7713	14.165	93,797	281
Gävle	60.667	17.1666	75,451	226
Örebro	59.2669	15.1965	155,989	467
		Population Sum:	2,934,285	
		Power Plants Production:	15854	
		Summation of demands:	8798	
		Average Power Per Capita:	3,000	

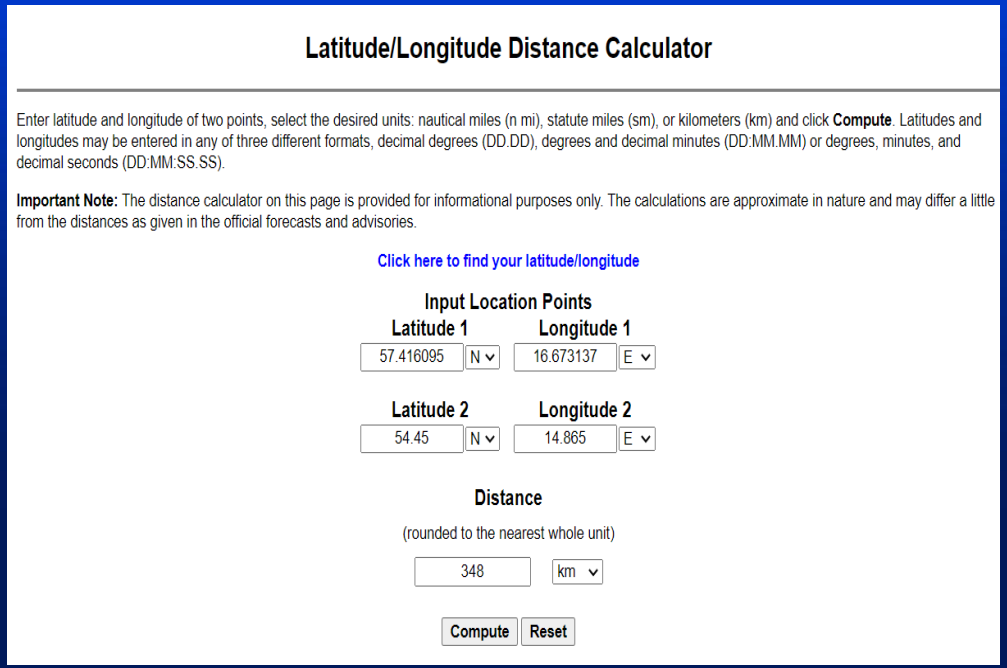
Transmission Lines Table

Line	Nodes	Length(km)	Bundles
1	1 4	842	2
2	2 4	886	2
3	3 4	74	2
4	4 6	282	2
5	5 6	450	2
6	6 10	94	1
7	7 10	220	2
8	10 19	400	2
9	10 14	124	1
10	8 11	546	2
11	9 11	52	1
12	11 12	545	2
13	11 13	95	2
14	11 14	95	1
15	11 15	125	2
16	13 15	119	2

Line	Nodes	Length(km)	Bundles
17	14 15	63	2
18	10 15	100	1
19	15 16	90	2
20	10 18	312	2
21	16 18	210	2
22	16 17	96	2
23	17 18	215	2
24	3 21	165	2
25	4 21	214	2
26	4 18	144	2
27	21 22	1015	2
28	20 21	210	1
29	18 21	235	2
30	18 20	272	2
31	18 20	220	2
32	20 23	878	2

Finding Transmission Lines Lengths

https://www.nhc.noaa.gov/gccalc.shtml



Latitude/Longitude Distance Calculator

Enter latitude and longitude of two points, select the desired units: nautical miles (n mi), statute miles (sm), or kilometers (km) and click **Compute**. Latitudes and longitudes may be entered in any of three different formats, decimal degrees (DD.DD), degrees and decimal minutes (DD:MM.MM) or degrees, minutes, and decimal seconds (DD:MM:SS.SS).

Important Note: The distance calculator on this page is provided for informational purposes only. The calculations are approximate in nature and may differ a little from the distances as given in the official forecasts and advisories.

Click here to find your latitude/longitude

Input Location Points

Latitude 1		Longitude 1	
57.416095	N	16.673137	E

Latitude 2		Longitude 2	
54.45	N	14.865	E

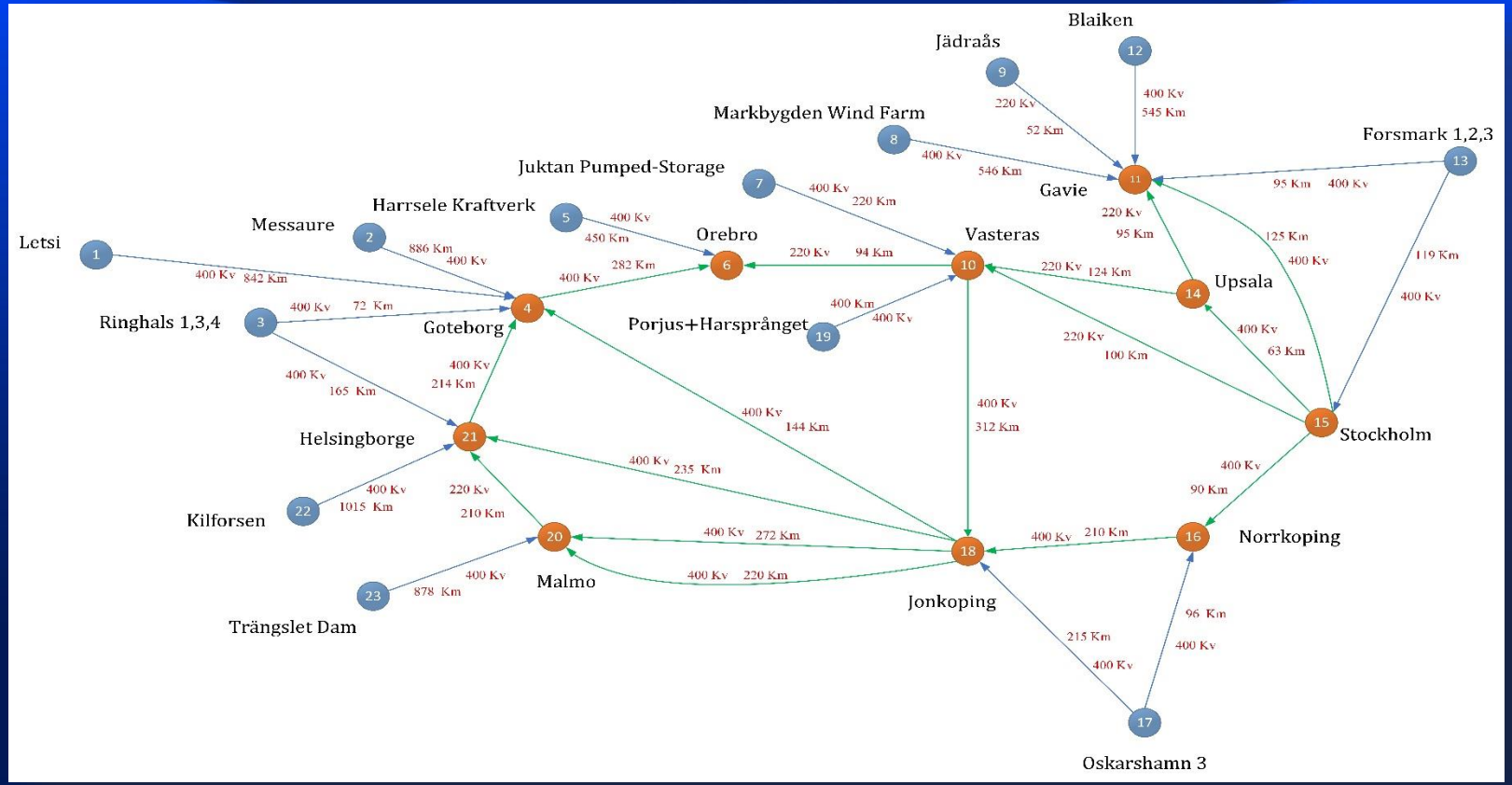
Distance

(rounded to the nearest whole unit)

348 km

Compute Reset

Dispatching Graph



Solution Algorithm

In [89]:
```
import mip
from mip import Model, xsum, minimize, BINARY
```

In [90]:
```
d = {4:1710, 6:467, 10:383, 11:226, 14:1129, 15:2917, 16:411, 18:281, 20:949, 21:325}
M = {1:456, 2:460, 3:3055, 5:223, 7:334, 8:4000, 9:203, 12:250, 13:3271, 17:1400, 19:1457, 22:415, 23:330}
```

In [91]:
```
I = [4, 6, 10, 11, 14, 15, 16, 18, 20, 21]
J = [1, 2, 3, 5, 7, 8, 9, 12, 13, 17, 19, 22, 23]
```

In [92]:
```
c = {(1,4):842, (2,4):886, (3,4):74, (4,6):282, (5,6):450, (6,10):94, (7,10):220, (10,19):400, (10,14):124, (8,11):546,
     (9,11):52, (11,12):545, (11,13):95, (11,14):95, (11,15):125, (13,15):119, (14,15):63, (10,15):100, (15,16):90,
     (10,18):312, (16,18):210, (16,17):96, (17,18):215, (3,21):165, (4,21):214, (4,18):144, (21,22):1015, (20,21):210,
     (18,21):235, (18,20):356, (20,23):878}
```

In [93]:
```
B = {(1,4):2, (2,4):2, (3,4):2, (4,6):2, (5,6):2, (6,10):1, (7,10):2, (10,19):2, (10,14):1, (8,11):2, (9,11):1, (11,12):2,
     (11,13):2, (11,14):1, (11,15):2, (13,15):2, (14,15):2, (10,15):1, (15,16):2, (10,18):2, (16,18):2, (16,17):2,
     (17,18):2, (3,21):2, (4,21):2, (4,18):2, (21,22):2, (20,21):1, (18,21):2, (18,20):2, (20,23):2}
```

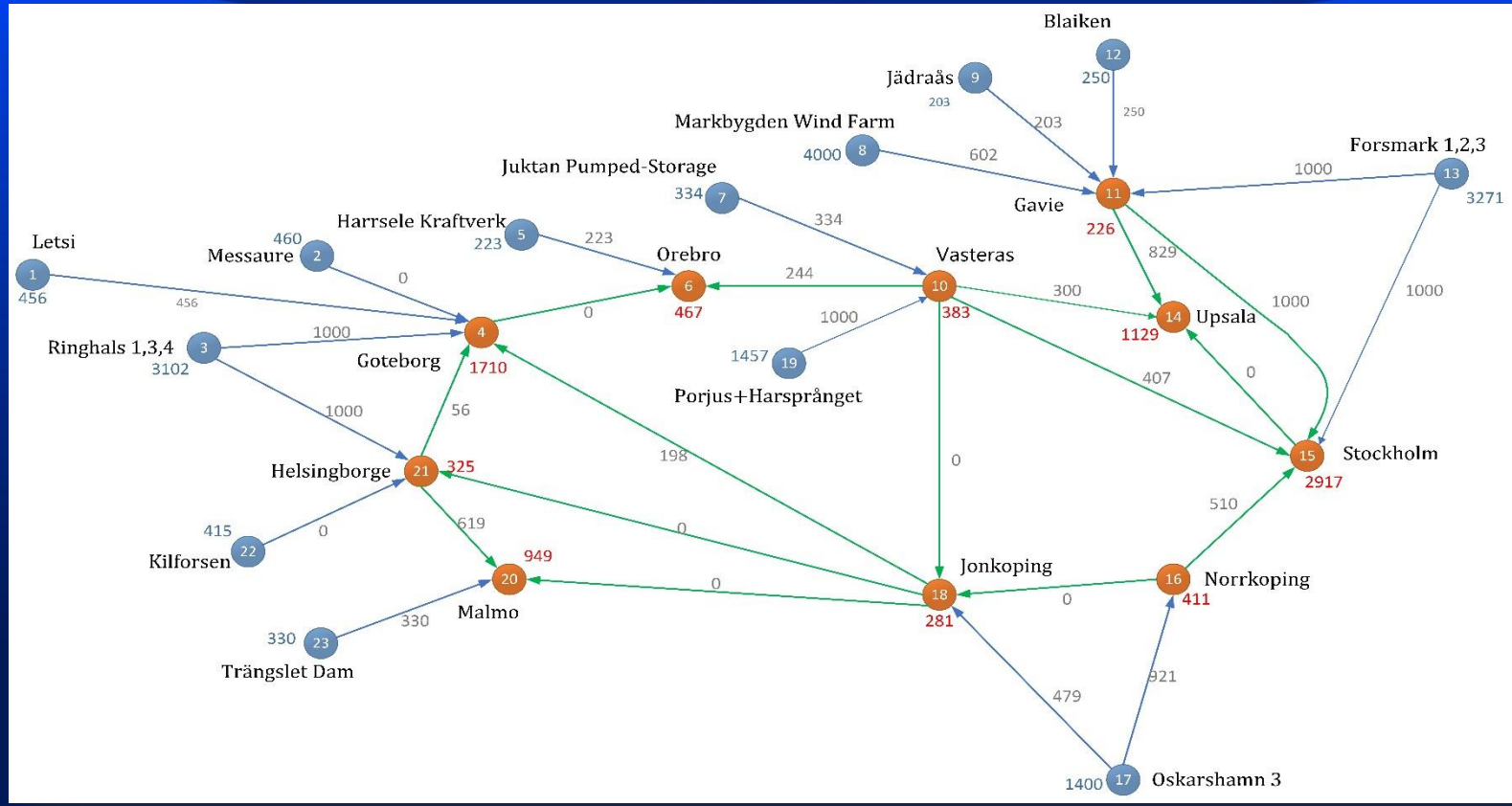
In [94]:
```
model = mip.Model()
x = {}
for i in I:
    for j in J:
        if ((i,j) in c or (j,i) in c) and not ((i,j) in x or (j,i) in x):
            if i > j:
                x[j,i] = model.add_var(var_type = mip.INTEGER)
            else:
                x[i,j] = model.add_var(var_type = mip.INTEGER)
```

Solution Algorithm Cont.

In [95]:
```
for i in I:
    for j in I:
        if ((i,j) in c or (j,i) in c) and not ((i,j) in x or (j,i) in x):
                x[j,i] = model.add_var(var_type = mip.INTEGER)
                x[i,j] = model.add_var(var_type = mip.INTEGER)
```

In [96]:
```
seen = {}
for i in I:
    constraints = []
    constraints2 = []
    for j in I + J:
        if (i,j) in x and (j,i) in x:
            if i < j:
                print(str(i) + "," + str(j))
                print("-" + str(j) + "," + str(i))
                constraints.append(x[i,j])
                constraints2.append(x[j,i])
            else:
                print(str(i) + "," + str(j))
                print("-" + str(j) + "," + str(i))
                constraints.append(x[i,j])
                constraints2.append(x[j,i])
        elif (i,j) in x:
            print(str(i) + "," + str(j))
            constraints.append(x[i,j])
        elif (j,i) in x:
            print(str(j) + "," + str(i))
            constraints.append(x[j,i])
    model.add_constr((xsum(constraints) - xsum(constraints2)) >= d[i])
    print("**********")
```

Output


Blaiken
12
250
250
Jädraås
9
203
203
Markbygden Wind Farm
4000
8
602
Forsmark 1,2,3
1000
13
3271
Juktan Pumped-Storage
334
7
334
Gavie
11
226
829
Harrsele Kraftverk
223
5
223
Letsi
460
Messaure
2
0
1
456
456
Orebro
6
467
244
Vasteras
10
383
300
1000
1000
1000
14
Upsala
1129
Ringhals 1,3,4
3
3102
1000
Goteborg
4
1710
0
1000
1457
19
Porjus+Harsprånget
407
0
56
198
0
Helsingborge
21
325
15
Stockholm
2917
510
415
22
0
Kilforsen
619
0
949
20
Malmo
Jonkoping
18
281
0
16
Norrkoping
411
0
330
23
330
Trängslet Dam
479
921
1400
17
Oskarshamn 3