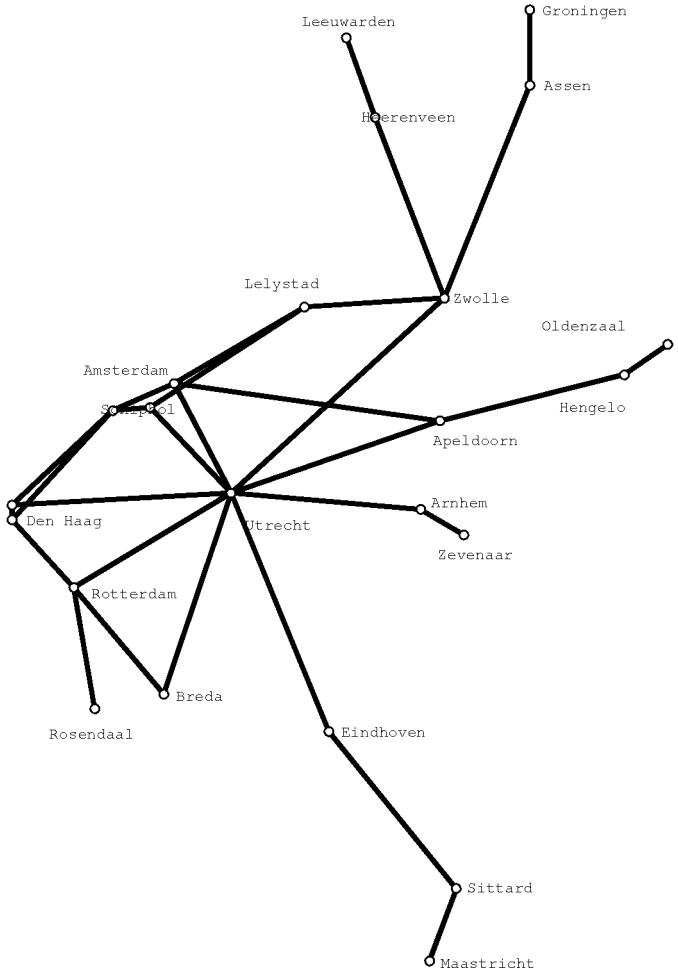
Excercise: Constrained-Shortest-Paths

For several exercises we use data for the Dutch intercity network (taken from a GAMS model by Michael Bussieck). The network looks as follows:



The data is contained in several files:

- ▷ *edges.dat* edges of the graph. They are directed and the data contains forward and backward directions (useful for ZIMPL).
- ▷ *times.dat* travel times for each edge (useful for ZIMPL).
- ▷ *costs.dat* costs each edge (useful for ZIMPL).
- ▷ *dutch.dat* all informations in one file (useful for C++ implementation).

We use the following abbreviations for station names:

Ah	Arnhem
Apd	Apeldoorn
Asd	Amsterdam CS
Asdz	Amsterdam Zuid WTC
Asn	Assen
Bd	Breda
Ehv	Eindhoven
Gn	Groningen
Gv	Den Haag HS
Gvc	Den Haag CS
Hgl	Hengelo
Hr	Heerenveen
Lls	Lelystad Centrum
Lw	Leeuwarden
Mt	Maastricht
Odzg	Oldenzaal Grens
Rsdg	Rosendaal Grens
Rtd	Rotterdam CS
Shl	Schiphol
Std	Sittard
Ut	Utrecht CS
Zl	Zwolle
Zvg	Zevenaar Grens

Exercise 1:

(a) Formulate the shortest path problem as an integer program.

(b) Use ZIMPL and SCIP to compute a shortest path from Groningen to Rotterdam in the dutch network (use times.dat as objective).

(c) Upgrade your model to solve constrained shortest path problems (important: values in times.dat are the objective coefficients and in cost.dat the weight coefficients).

(d) Compute the fastest path from Groningen to Rotterdam with a maximal weight of 20.000.

(e) Solve the LP-relaxation of your model.

Exercise 2: Implement a dynamic program to solve the constrained shortest path problem. Use rcsp.cpp as a framework for your implementation. You only have to implement the function "void cspDP()" in rcsp.cpp. Hints for compiling and executing rcsp.cpp are in the file.