

Mathematical Programming: Modeling and Applications

Giacomo Nannicini

`giacomon@lix.polytechnique.fr`

Queens on a Chessboard

- We have a standard chessboard (8×8)
- How many queens can there be so that no queen is under threat by the others? Where should we position them?
- Formulate a program where the size of the (square) chessboard is a parameter, and we want to find the maximum number and position of the queens

Solution

- Parameters: n = number of cells on one side of the chessboard
- Variables: $x_{ij} = 1$ if there is a queen in position (i, j) , 0 otherwise
- Objective function: $\max \sum_{i,j \leq n} x_{ij}$

Solution: Constraints

- At most one queen per row: $\forall i \leq n \left(\sum_{j \leq n} x_{ij} \leq 1 \right)$
- At most one queen per column: $\forall j \leq n \left(\sum_{i \leq n} x_{ij} \leq 1 \right)$
- At most one queen per NW-SE diagonal: $\forall i, j \leq n$

$$\sum_{h \leq n: h \leq i, h \leq j} x_{(i-h)(j-h)} + \sum_{h \leq n: h+i \leq n, h+j \leq n} x_{(i+h)(j+h)} \leq 1$$

- At most one queen per SW-NE diagonal: $\forall i, j \leq n$

$$\sum_{h \leq n: h \leq i, h+j \leq n} x_{(i-h)(j+h)} + \sum_{h \leq n: h+i \leq n, h \leq j} x_{(i+h)(j-h)} \leq 1$$

Solution: Model

```
# queens.mod
param n >= 0, default 8;
set N := 1..n;
var x{N,N} binary;

maximize queens : sum{i in N, j in N} x[i,j];

subject to rows {i in N} : sum{j in N} x[i,j] <= 1;
subject to cols {j in N} : sum{i in N} x[i,j] <= 1;
subject to diagNW {i in N, j in N} :
    sum{h in N : h < i and h < j} x[i-h,j-h] +
    sum{h in N : h+i<=n and h+j<=n} x[i+h,j+h] <= 1;
subject to diagSW {i in N, j in N} :
    sum{h in N : h < i and h+j<=n} x[i-h,j+h] +
    sum{h in N : h+i<=n and h < j} x[i+h,j-h] <= 1;
```

Solution: Run File

```
# queens.run  
model queens.mod;  
option solver cplex;  
solve;  
display queens;  
display x;
```

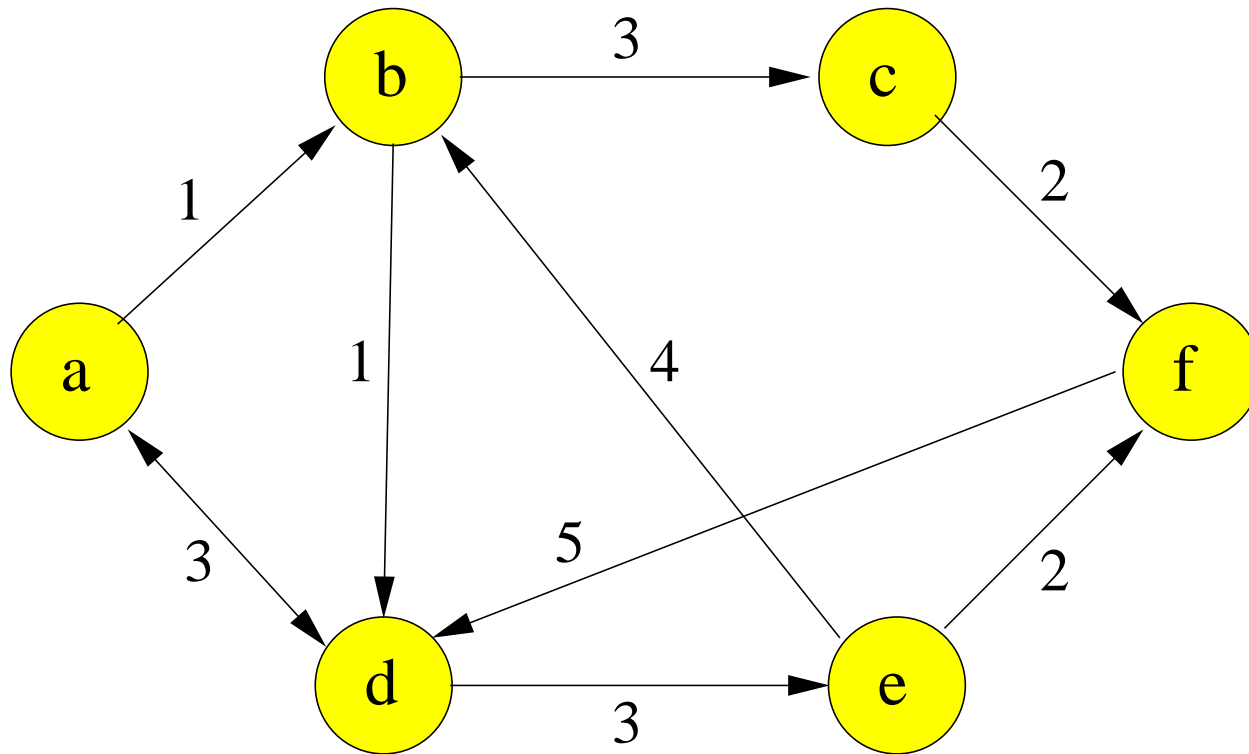
Solution: Output

```
ILOG AMPL 10.100, licensed to "ecolepolytechnique-palaiseau".  
AMPL Version 20060626 (Linux 2.6.9-5.ELsmp)  
ILOG CPLEX 10.100, licensed to "ecolepolytechnique-palaiseau"  
CPLEX 10.1.0: optimal integer solution; objective 8  
146 MIP simplex iterations  
0 branch-and-bound nodes  
queens = 8
```

```
x [*,*]  
:   1   2   3   4   5   6   7   8   :=  
1   0   0   0   0   1   0   0   0  
2   0   0   0   0   0   0   1   0  
3   1   0   0   0   0   0   0   0  
4   0   0   1   0   0   0   0   0  
5   0   0   0   0   0   0   0   1  
6   0   0   0   0   0   1   0   0  
7   0   0   0   1   0   0   0   0  
8   0   1   0   0   0   0   0   0  
;
```

The Shortest Path Problem

- We have a graph $G = (V, A)$ and a cost $c : A \rightarrow \mathbb{R}^+$



- We want to compute the shortest path between two nodes in the graph (source and sink/target)

Network Flow

- This kind of problems can be modeled using network flows
- Main idea: see the graph as interconnected pipes, each one with a cost
- We insert units of flow at some nodes of the network, and want to receive some units of flow at selected nodes
- Often, there is a maximum capacity for each arc
- Variables: amount of flow on each arc
- Subject to flow conservation: at each node, incoming flow = outgoing flow; if the node produces or receives some flow, then incoming flow + inserted flow = outgoing flow + received flow

Network Flow: Motivation

- The shortest path problem is a very simple example of network flow application
- One more example: node-disjoint paths (arises when building fault-tolerant networks)
- Idea: put a maximum capacity of 1 on each arc; then, if we send k units of flow from the source to the destination, the solution of the network flow problem will consist in k node-disjoint paths
- The formulation is very flexible!

Incidence Matrix

- How do we represent graph connectivity?
- One (typical) way is a $|V| \times |A|$ incidence matrix M
- Each element m_{ij} is 1 if arc j is an outgoing arc for node i , -1 if arc j is an incoming arc for node i , 0 otherwise
- Therefore, on each row (which corresponds to an arc) there is exactly one -1 and one 1

Network Flow: Shortest Path



- The Shortest Path Problem between two nodes of the network can be modeled with this paradigm
- We insert one unit of flow at the source, and want to receive one unit of flow at the destination
- Variables: in this case, the flow on each arc is either zero or one
- Write a model for the SPP and a data file for the graph showed above
- Write a run file that computes and displays the shortest path between all pairs of nodes in the graph

Solution: Formulation

- Sets: vertices V , arcs $A \subset V \times V$
- Parameters: cost c_{jk} of each arc $(j, k) \in A$, incidence matrix $M \in \{-1, 0, 1\}^{|V| \times |A|}$, demands d_i for each node $i \in V$
- Variables: x_{jk} units of flow on arc (j, k)
- Objective: $\min \sum_{(j,k) \in A} c_{jk} x_{jk}$
- Subject to: flow conservation

$$\forall i \in V \left(\sum_{(j,k) \in A} m_{ijk} x_{jk} = d_i \right)$$

Solution: Model

```
set V;  
set A within V cross V;  
  
param cost{A};  
param incidence{V,A};  
param demands{V};  
  
var x{A} >= 0;  
  
minimize pathcost: sum{(i,j) in A} cost[i,j]*x[i,j];  
  
subject to conservation{i in V}:  
    sum{(j,k) in A} incidence[i,j,k]*x[j,k] = demands[i];
```

Solution: Data

```
set V := a b c d e f;
```

```
set A := (a,b) (a,d) (b,c) (b,d) (c,f) (d,a) (d,e) (e,b) (e,f) (f,d);
```

```
param cost:=
```

```
a b 1
```

```
a d 3
```

```
b c 3
```

```
b d 1
```

```
c f 2
```

```
d a 3
```

```
d e 3
```

```
e b 4
```

```
e f 2
```

```
f d 5;
```

```
param demands:=
```

```
a 1
```

```
b 0
```

```
c 0
```

```
d 0
```

```
e 0
```

```
f -1;
```

```
param incidence(tr): a b c d e f:=
```

```
a b 1 -1 0 0 0 0
```

```
a d 1 0 0 -1 0 0
```

```
b c 0 1 -1 0 0 0
```

```
b d 0 1 0 -1 0 0
```

```
c f 0 0 1 0 0 -1
```

```
d a -1 0 0 1 0 0
```

```
d e 0 0 0 1 -1 0
```

```
e b 0 -1 0 0 1 0
```

```
e f 0 0 0 0 1 -1
```

```
f d 0 0 0 -1 0 1;
```

Solution: Run File

```
option solver cplex;
model dijkstra.mod;
data dijkstra.dat;
for {i in V}{
  for {j in V diff {i}}{
    for {h in V}{
      if (h = i) then let demands[h]:=1;
      else if (h = j) then let demands[h]:= -1;
      else let demands[h]:= 0;
    }
  }
  solve;
  printf "Shortest path between %s and %s: ",i,j > output_dijkstra.txt;
  for {h in V}{
    for {(u,v) in A: u = h}{
      if (x[u,v] > 0) then printf "%s %s, ",u,v > output_dijkstra.txt;
    }
  }
  printf "with cost %d \n", pathcost > output_dijkstra.txt;
}
}
```

Solution: Output

Shortest path between a and b: a b, with cost 1
Shortest path between a and c: a b, b c, with cost 4
Shortest path between a and d: a b, b d, with cost 2
Shortest path between a and e: a b, b d, d e, with cost 5
Shortest path between a and f: a b, b c, c f, with cost 6
Shortest path between b and a: b d, d a, with cost 4
Shortest path between b and c: b c, with cost 3
Shortest path between b and d: b d, with cost 1
Shortest path between b and e: b d, d e, with cost 4
Shortest path between b and f: b c, c f, with cost 5
[...]
Shortest path between f and a: d a, f d, with cost 8
Shortest path between f and b: a b, d a, f d, with cost 9
Shortest path between f and c: a b, b c, d a, f d, with cost 12
Shortest path between f and d: f d, with cost 5
Shortest path between f and e: d e, f d, with cost 8

Unimodularity

- The flow variables x_{jk} were not constrained to be integer
- Yet, in the solution they were binary
- This is because the incidence matrix M is unimodular, i.e. the determinant of all its invertible square submatrices is either $+1$ or -1
- Nice property of problems with constraint $Mx = d$ with M unimodular: if M and d have integer elements, then all solutions to the LP are integer
- Therefore, network flow problems are easy to solve!