Department of Mathematics and Computer Science
University of Southern Denmark, Odense

May 16, 2018
Marco Chiarandini

DM559/DM545 – Linear and integer programming

Sheet 9, Spring 2018 [pdf format]

Solution:
Contains Solutions!

Exercise 1* Problem of Representatives

A town has r residents $R_1, R_2, \ldots, R_r$; q clubs $C_1, C_2, \ldots, C_q$; and p political parties $P_1, P_2, \ldots, P_p$. Each resident is a member of at least one club and can belong to exactly one political party. Each club must nominate one of its members to represent it on the town's governing council so that the number of council members belonging to the political party P_k is at most u_k. Is it possible to find a council that satisfies this "balancing" property?
Show how to formulate this problem as a maximum flow problem.

Solution:
[AMO] at library bookshelf pages 170-176.

Exercise 2* Scheduling on Uniform Parallel Machines

We consider scheduling a set J of jobs on M uniform parallel machines. Each job $j \in J$ has a processing requirement p_j (denoting the number of machine days required to complete the job), a release data r_j (representing the beginning of the day when job j become available for processing), and a due date $d_j \geq r_j + p_j$ (representing the beginning of the day by which the job must be completed). We assume that a machine can work on only one job at a time and that each hob can be processed by at most one machine at a time. However we allow preemptions (ie, we can interrupt a job and process it on different machines on different days). The scheduling problem is to determine a feasible schedule that completes all jobs before their due dates or to show that no such schedule exists.
Formulate the feasible scheduling problem as a maximum flow problem.

Solution:
[AMO] pages 170-176.

Exercise 3* Tanker Scheduling Problem

A steamship company has contracted to deliver perishable goods between several different origin-destination pairs. Since the cargo is perishable the customers have specified precise dates (ie, delivery dates) when the shipments must reach their destinations. (The cargoes may not arrive early or late). The steamship company wants to determine the minimum number of ships needed to meet the delivery dates of the shiploads.
Formulate this problem as a maximum flow problem modeling the following example with four shipments. Each shipment is a full shipload with the characteristics shown in Table 1. For example, as specified by the first row in this figure, the company must deliver one shipload available at port A and destined for port C on day 3.

Solution:
[AMO] pages 170-176.

Exercise 4*

Given the Network in Figure 1, determine the max flow and indicate the min cut.
You can solve the problem in Guroby Python. At the exam you may be asked similar questions and hence having a code template ready might be very useful.

shipment	origin	destination	delivery date
1	Port A	Port C	3
2	Port A	Port C	8
3	Port B	Port D	3
4	Port B	Port C	6

	C	D
A	3	2
B	2	3

	A	B
C	2	1
D	1	2

Table 1: Data for the tanker scheduling problem: Left shipment characteristics; Center, shipment transit times; Right return times.

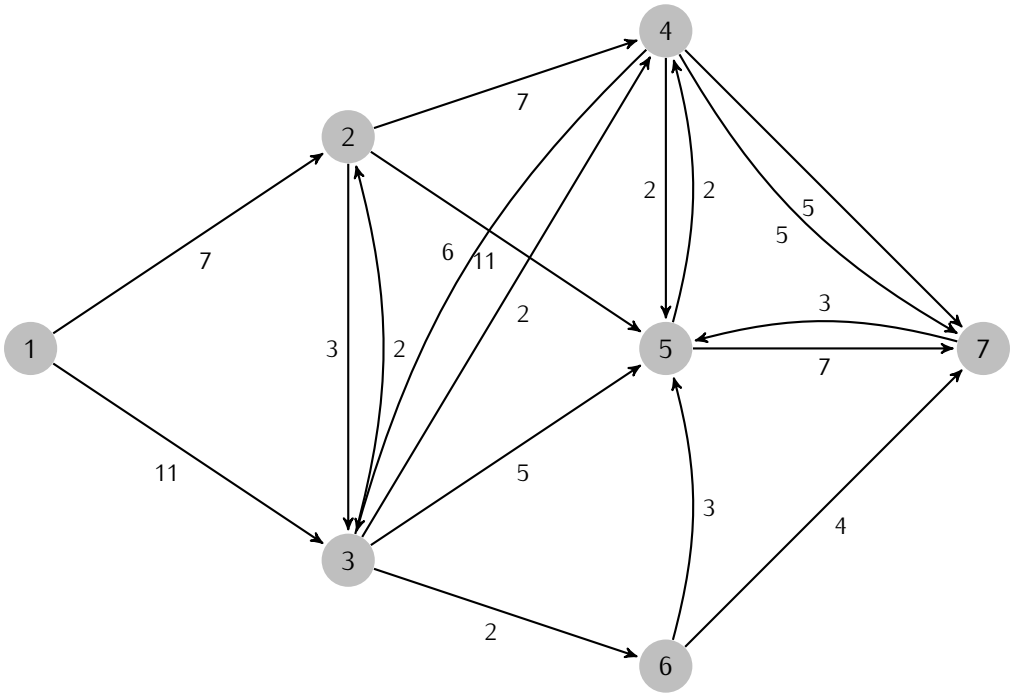


Figure 1: Find the maximum flow from 1 to 7. Numbers on arcs are capacity values.

[In preparation of the digital exam, at the end of this document you find the excerpt of latex code to produce the Figure 1. You can use it to experiment whether its use is fast enough for an exam session.]

Solution:

To implement the mincost flow model in Gurobi Python the best starting point is perhaps the mulit-commodity flow example at Gurobi. In particular, it is worth observing the use of the Gurobi data type tuplelist in the flow conservatioon constraints.
It is first necessary to formulate the max flow problem as a min cost flow problem. This has been done in class and it is available from the slides.

```
#!/usr/bin/python

from gurobipy import *
import networkx as nx
import matplotlib.pyplot as plt
from networkx.drawing.nx_pydot import write_dot


nodes = ['a','b','c','d','e','f','g','h']

arcs, capacity = multidict({
('a','b'): 15,
('a','c'): 5,
('a','d'): 10,
('d','c'): 4,
```

```
('c','b'): 4,
('d','g'): 9,
('d','f'): 15,
('c','f'): 8,
('b','e'): 30,
('e','c'): 6,
('g','f'): 15,
('f','e'): 15,
('g','h'): 10,
('f','h'): 10,
('e','h'): 10,
('h','a'): GRB.INFINITY})

arcs = tuplelist(arcs)


cost = {
('a','b'): 0,
('a','c'): 0,
('a','d'): 0,
('d','c'): 0,
('c','b'): 0,
('d','g'): 0,
('d','f'): 0,
('c','f'): 0,
('b','e'): 0,
('e','c'): 0,
('g','f'): 0,
('f','e'): 0,
('g','h'): 0,
('f','h'): 0,
('e','h'): 0,
('h','a'): -1}


inflow={
'a':0,
'b':0,
'c':0,
'd':0,
'e':0,
'f':0,
'g':0,
'h':0
}


G = nx.DiGraph()
G.add_nodes_from(nodes)
G.add_edges_from(arcs)
#G.add_weighted_edges_from(arcs,cost)

pos=nx.spectral_layout(G)
# pos = nx.kamada_kawai_layout(G)
nx.draw(G, pos=pos, with_labels=True)
#nx.draw_networkx_edge_labels(G,pos,edge_labels={e:G.edge[e[0]][e[1]]["capacity"] for e in
     G.edges()})
plt.show()
plt.savefig("graph.png")

write_dot(G,"graph.dot")

# Create optimization model
```

```
m = Model('netflow')

# Create variables
flow = {}
for i,j in arcs:
    flow[i,j] = m.addVar(ub=capacity[i,j], obj=cost[i,j],vtype=GRB.CONTINUOUS,
                         name='flow_%s_%s' % (i, j))
# Alternative way
# flow = m.addVars(arcs, obj=cost, name="flow")

m.modelSense=GRB.MINIMIZE
# Flow conservation constraints
for j in nodes:
    m.addConstr(
        quicksum(flow[i,j] for i,j in arcs.select('*',j))
        - quicksum(flow[j,k] for j,k in arcs.select(j,'*'))==inflow[j],
                 'node_%s' % ( j))

# Compute optimal solution
m.optimize()
m.write("mincost.lp")
# Print solution
if m.status == GRB.status.OPTIMAL:
    solution = m.getAttr('x', flow)
    print('\nOptimal flow:')
    for i,j in arcs:
        if solution[i,j] > 0:
            print('%s -> %s: %g' % (i, j, solution[i,j]))
```

An alternative way is to solve the max flow problem with dedicated algorithms. The networkx module in Python has several of these algorithms implemented and it can also easily plot networks and generate graphs.
Here is an example of use for the network in the next exercise:

```
import networkx as nx
G = nx.DiGraph()
G.add_edge('a','b',capacity= 15)
G.add_edge('a','c',capacity= 5)
G.add_edge('a','d',capacity= 10)
G.add_edge('d','c',capacity= 4)
G.add_edge('c','b',capacity= 4)
G.add_edge('d','g',capacity= 9)
G.add_edge('d','f',capacity= 15)
G.add_edge('c','f',capacity= 8)
G.add_edge('b','e',capacity= 30)
G.add_edge('e','c',capacity= 6)
G.add_edge('g','f',capacity= 15)
G.add_edge('f','e',capacity= 15)
G.add_edge('g','h',capacity= 10)
G.add_edge('f','h',capacity= 10)
G.add_edge('e','h',capacity= 10)

F = nx.ford_fulkerson_flow(G, 'a', 'h')
for u, v in G.edges_iter():
    print('(%s, %s) %.2f' % (u, v, F[u][v]))
```

Exercise 5* Directed Chinese Postman Problem

Suppose a postman has to deliver mail along all the streets in a small town. Assume furthermore that on one-way streets the mail boxes are all on one side of the street, whereas for two-way streets, there are mail boxes on both sides of the street. For obvious reasons the postman wishes to minimize the

distance he has to travel in order to deliver all the mail and return home to his starting point. Show how you can solve this problem using minimum cost flows. A similar model can be formulated for the Snow Plow problem (http://city.temeda.com/) or the Salt Spreding problem.

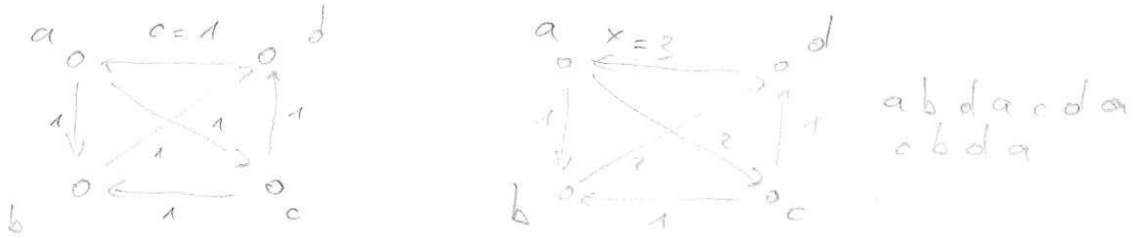
Solution:

The solution to this problem can be found on page 174 of J. Bang-Jensen and G. Gutin. Digraphs: Theory, Algorithms and Applications, Springer London, 2009 http://dx.doi.org/10.1007/978-1-84800-998-1_4.
In short, the solution to the Chinese postman problem is a min cost flow that traverses each arc at least once in a network where arcs are streets and nodes are street instersections.
We must assume that the graph is strongly connected otherwise there is no solution (any closed walk is strongly connected.)
A digraph where there exists a walk that visits arcs exactly once is called Eulerian.

$$N = (V, A, \mathbf{l}, \mathbf{u} = \infty, \mathbf{c})$$

$$x_{ij} = \#\text{ of times arc is used}$$

The cost os a min cost circulation in N equals the cost of a Chinese postman walk in D.



Exercise 6 [DT97]

Suppose that in a minimum cost flow problem restrictions are placed on the total flow leaving a node k, i.e.

$$\underline{\theta}_k \leq \sum_{(k,j)\in E} x_{kj} \leq \overline{\theta}_k$$

Show how to modify these restrictions to convert the problem into a standard cost flow problem.

Solution:

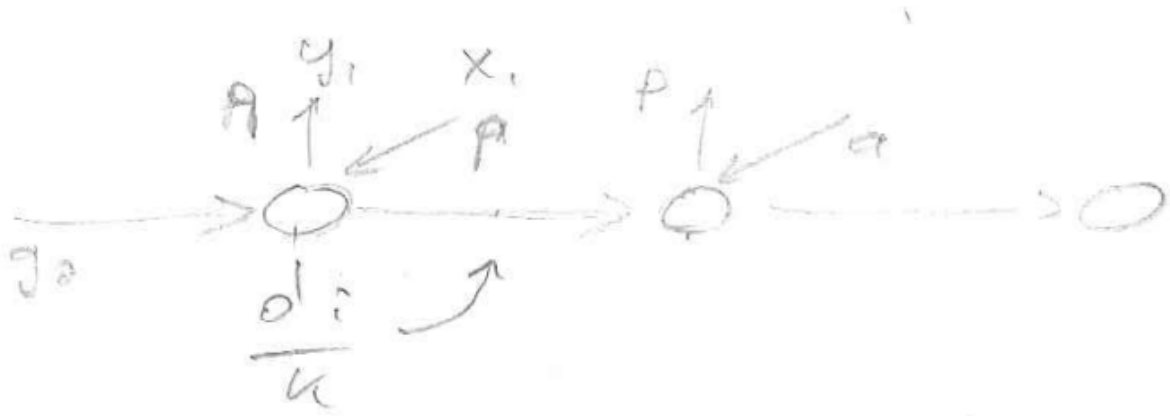
Bounds on nodes are not possible in our model of min cost flow. However, we can transform the network by splitting the vertex in two and introducing an arc between the new vertices with the given bounds.

Exercise 7

The production plan of a factory for the next year is to produce d_t units of product per month t, $t = 1, \ldots, 12$. Each worker can produce k units of product in a month. The monthly salary is equal to s. Employing and firing personnel has costs: precisely, employing one person costs p while firing one costs q. Assuming that initially there are g_0 workers, determine the number of workers that must be present during every month such that the demand is always satisfied and the overall costs of salary, employment, and firing are minimized.

Solution:

It is possible to model the employment and firing of workers as a flow in a network.



Exercise 8

The Classical airline crew scheduling problem is described as follows. A flight segment is a nonstop flight between two airports with particular departure/arrival times and airplane requirements. Crew schedules are determined for each fleet (airplanes of a particular type) separately. First a subset of *rotations (or rosters)* are identified. Rotations are collections of flight segments that satisfy aviation regulations, company regulations and labor union rules and that are meant to be served by a single crew. They have typically duration of 2-3 days. Then, crews are assigned to the selected rotations, often by a bidding system.
The cost of selecting a particular rotation includes flight time of the crew in that rotation, lodging and per day expenses, cost of transporting crews to their next departure city in their next flight segment (they are not serving on these flight segments), etc. Let c_j denote the cost of selecting rotation $j \in \{1, \ldots, n\}$. Suppose there are m flight segments, then a rotation j is given by an incidence vector of flight segments in that rotation, i.e., for rotation j, $\mathbf{a}_j$ is an m-dimensional vector with $a_{ij} = 1$ if flight segment i is on rotation j and 0 otherwise. Formulate the problem of selecting the rotations that minimize the total cost such that all flight segments are served, and each flight segment is served by exactly one rotation.

Solution:

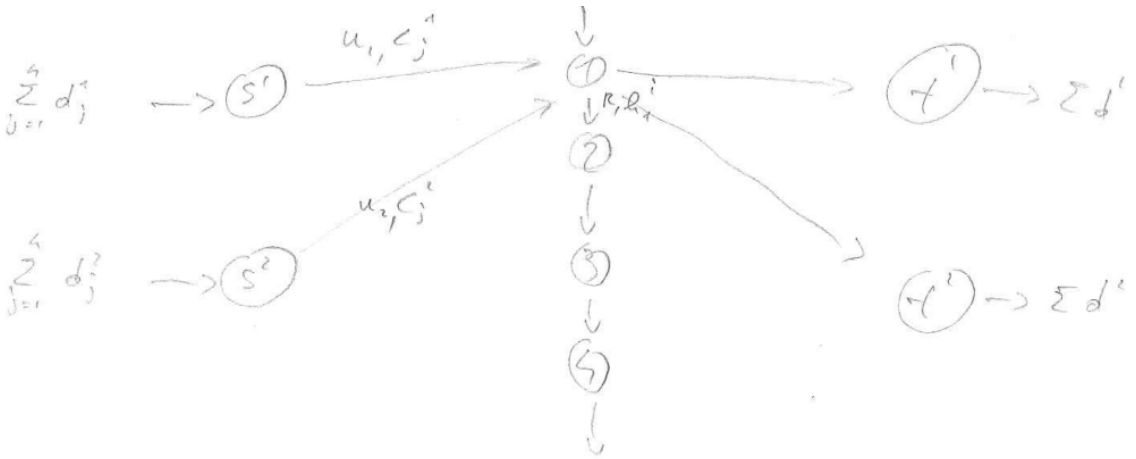
This is an example of application of the set partitioning model.

Exercise 9 Warehousing of Seasonal Products

A company manufactures multiple products. The products are seasonal with demand varying weekly, monthly, or quarterly. To use its work-force and capital equipment efficiently, the company wishes to "smooth" production, storing pre-season production to supplement peak-season production. The company has a warehouse with fixed capacity R that it uses to store all the products it produces. Its decision problem is to identify the production levels of all the products for every week, month, or quarter of the year that will permit it to satisfy the demands incurring the minimum possible production and storage costs.
We can represent this warehousing problem as a relevant generalization of the min cost network flow problem encountered in the course.
For simplicity, consider a situation in which the company makes two products and then it needs to schedule its production for each of the next four quarters of the year. Let d_j^1 and d_j^2 denote the demand for products 1 and 2 in quarter j. Suppose that the production capacity for the jth quarter is u_j^1 and u_j^2, and that the per unit cost of production for this quarter is c_j^1 and c_j^2. Let h_j^1 and h_j^2 denote the storage (holding) costs per unit of the two products from quarter j to quarter $j+1$.
Represent graphically the network in the two products four periods case and write the Linear Programming formulation of the problem. Which network flows problem models this application? If all input data are integer, will the solution be integer?

Solution:

We can model this problem as a multicommodity flow in a network. The network has 4 nodes, one for each quarter, two target nodes for each quarter (one per product), two nodes for each plant and for each quarter (one per product) and two global sources for the two products.
See page 655 of [AMO].



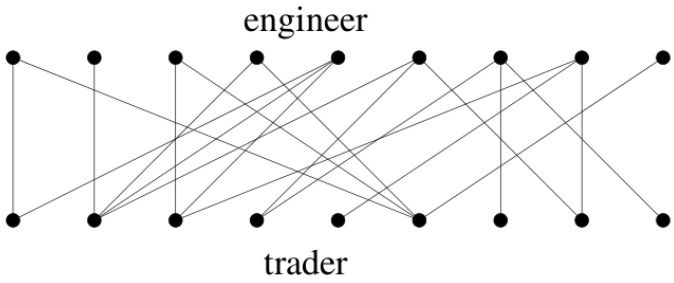


Figure 2:

Exercise 10

A managing director has to launch the marketing of a new product. Several candidate products are at his disposal and he has to choose the best one. Hence, he let each of these products be analysed by a team made of an engineer and a trader who write a review together. The teams are made along the graph in Figure 2; each edge corresponds to a product and its endvertices to the engineer and trader examining it.

a) How many people at least does the managing director gather in order to have the report on all the products? (The report can be given by either the engineer or the trader.)

b) Assuming now that the report must be done jointly by an engineering and a trader, and that each engineer and trader can be occupied with only one candidate product, give a polynomial time algorithm to identify which products will for sure not have the possibility to obtain a report.

Solution:

a) This is an application of the vertex cover problem and its strong duality with maximum matching.
b) This can be done by finding all maximum matching of the graph. The edges that are never in a matching are those that will be never reviewed. We could solve $|E|$ linear programs formulations of the max matching problem for bipartite graphs in each of which a different edge is enforced to be in the solution. Other, more efficient methods based on direct algorithms exist.

References

[DT97] George B. Dantzig and Mukund N. Thapa. *Linear Programming*. Springer, 1997.

Appendix

```
\documentclass{article}
\usepackage{tikz}
\usetikzlibrary{arrows,decorations.pathmorphing,backgrounds,positioning,fit,matrix}

\begin{document}
\begin{tikzpicture}[scale=1.4, auto,swap]
\tikzstyle{vertex}=[circle,fill=black!25,minimum size=20pt,inner sep=0pt]
\tikzstyle{selected vertex} = [vertex, fill=red!24]
\tikzstyle{edge} = [draw,thick,-]
\tikzstyle{arc} = [draw,thick,->,shorten >=1pt,>=stealth']
\tikzstyle{weight} = [font=\small]
\tikzstyle{selected edge} = [draw,line width=5pt,-,red!50]
\tikzstyle{ignored edge} = [draw,line width=5pt,-,black!20]

    % First we draw the vertices
    \foreach \pos/\name in {{(0,2)/a}, {(2,0)/b}, {(2,2)/c},
                        {(2,4)/d}, {(4,0)/e}, {(4,2)/f}, {(4,4)/g}, {(6,2)/h}}
        \node[vertex] (\name) at \pos {$\name$};
    % Connect vertices with edges and draw weights
    \foreach \source/ \dest /\weight in {
      a/b/15,
      a/c/5,
      a/d/10,
      d/c/4,
      c/b/4,
      d/g/9,
      d/f/15,
      c/f/8,
      b/e/30,
      e/c/6,
      g/f/15,
      f/e/15,
      g/h/10,
      f/h/10,
      e/h/10}
        \path[arc] (\source) -- node[weight] {$\weight$} (\dest);
\end{tikzpicture}
```