ProblemSet 4 – Integer programs and max flow

George McNinch

2024-02-23

1. One man's trash

   You are at a yard sale, and have spied four crates of goods. You've estimated the value of each crate; these values are listed as **actual value** in the following table.

   The owner has no idea what these items are worth, and is selling them for far less than they are worth; the prices being asked are listed as **sales price** in the following table).

   Finally, the *weight* of each of the crates is listed in the table, as well.

crate	actual value	sales price	weight in kg
A	$ 5000	$ 24	75.5
B	$ 600	$ 76	2.7
C	$ 3500	$ 43	3.3
D	$ 6000	$ 754	6.7

   You realize that you can purchase these crates and sell them at a much higher mark up. However, you walked to the yard sale and can only buy what you can carry on your person.

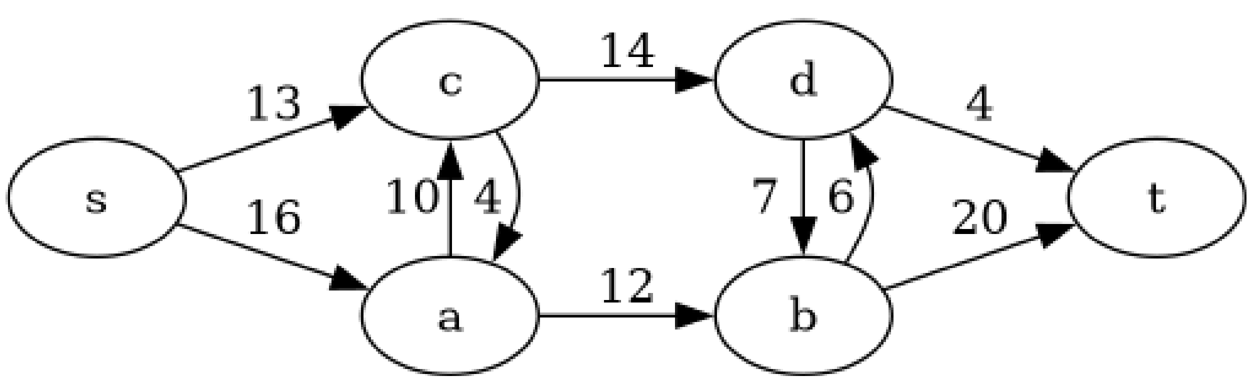
   You have 800 dollars, and you and your friend together can carry an estimated 85 kg.

   Fortunately, you have identified this as an integer programming problem!

   a. Describe an integer linear program which models the situation (think carefully about what values the variables can take).
   b. Use the branch and bound algorithm to find the optimal solution, explaining your choices for which variables to branch on and where to prune the tree.
   c. Draw the branch and bound tree for your solution.

   (**Hint**: Note that you should use `linprog` to solve the relaxed linear program, initially with your variables constrained between 0 and 1).

2. Consider the following directed graph (see below for the code producing this graph).



a. Find the `minimum cut` value for this weighted directed graph.

   (Recall that this means to consider all possible partitions of the nodes into an s-group and a t-group.

   To identify such a partition, it is enough to indicate the s-group. For example, s, a and c together form a possible s-group. The edge-cuts required to form this s-group involve the edges $c \to d$ and $a \to b$; thus the cut-value for this partition is $14 + 12 = 26$.

   Make a list all possible s-groups and indicate the corresponding cut-values. Remember that we are only interested in partitions that arise "from cuts" – thus, an s-group should be "connected". And remember that cut value only involve capacities of edges $u \to v$ where u is in the s-group and v is in the t-group (you would not also count the capacity of an edge $v \to u$ if it exists).

b. By strong duality, you now know the `maximum flow` value for the graph. Does strong duality tell you how to find a flow which achieves this value? Why or why not?

c. Suppose that the capacity on the edge $d \to t$ is increased from its current value of 4. By how much must this capacity change in order to change the `min cut` configuration?

d. Would increasing the capacity for the edge $b \to t$ from its current value of 20 affect the maximum flow value? Why or why not?

e. Explain whether or not the following statement seems reasonable, and why: "The `min cut` configuration (i.e. the edges involved in the `min cut`) reflects the maximal bottleneck in the system."

Here is the code used to produce the graph; you can download it here.

```
from graphviz import Digraph

dg = Digraph()
#dg = Digraph(engine='neato')

dg.attr(rankdir='LR')

V = [ 's', 'a', 'b', 'c', 'd', 't' ]

weights = { ('s','a'):   16,
            ('s','c'):   13,
            ('c','a'):    4,
            ('a','c'):   10,
            ('a','b'):   12,
            ('c','d'):   14,
            ('d','b'):    7,
            ('b','d'):    6,
            ('b','t'):   20,
            ('d','t'):    4
          }

c.node('t')

with dg.subgraph() as c:
    c.attr(rank='same')
    for x in ['a','c']:
        c.node(x)

with dg.subgraph() as c:
    c.attr(rank='same')
    for x in ['b','d']:
        c.node(x)

c.node('s')
```

```python
for (f,t) in weights.keys():
    wt = f"{weights[(f,t)]}"
    dg.edge(f,t,wt)

dg.filename='PS4--graph'
dg.format='png'
dg.render()
```