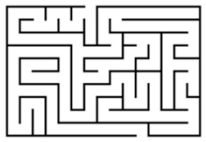
Flow Network

Analysis and implementation

jasonMaynard

fall_13



Given Flow Network Problem

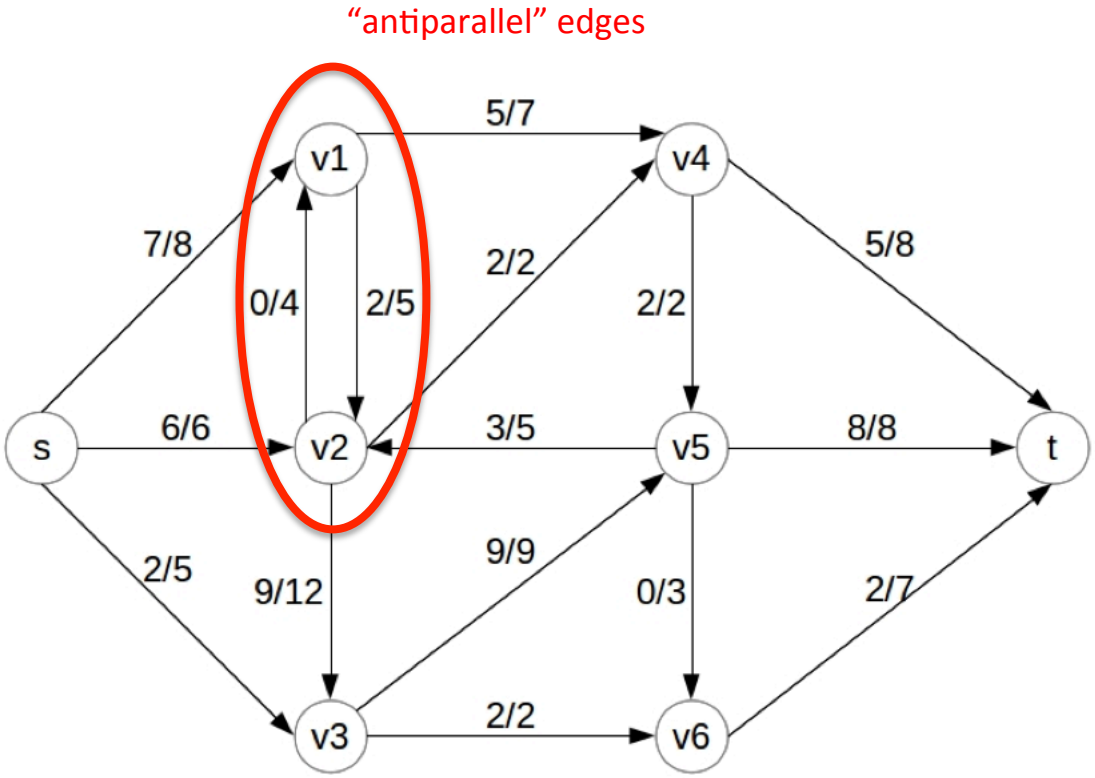
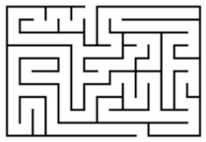
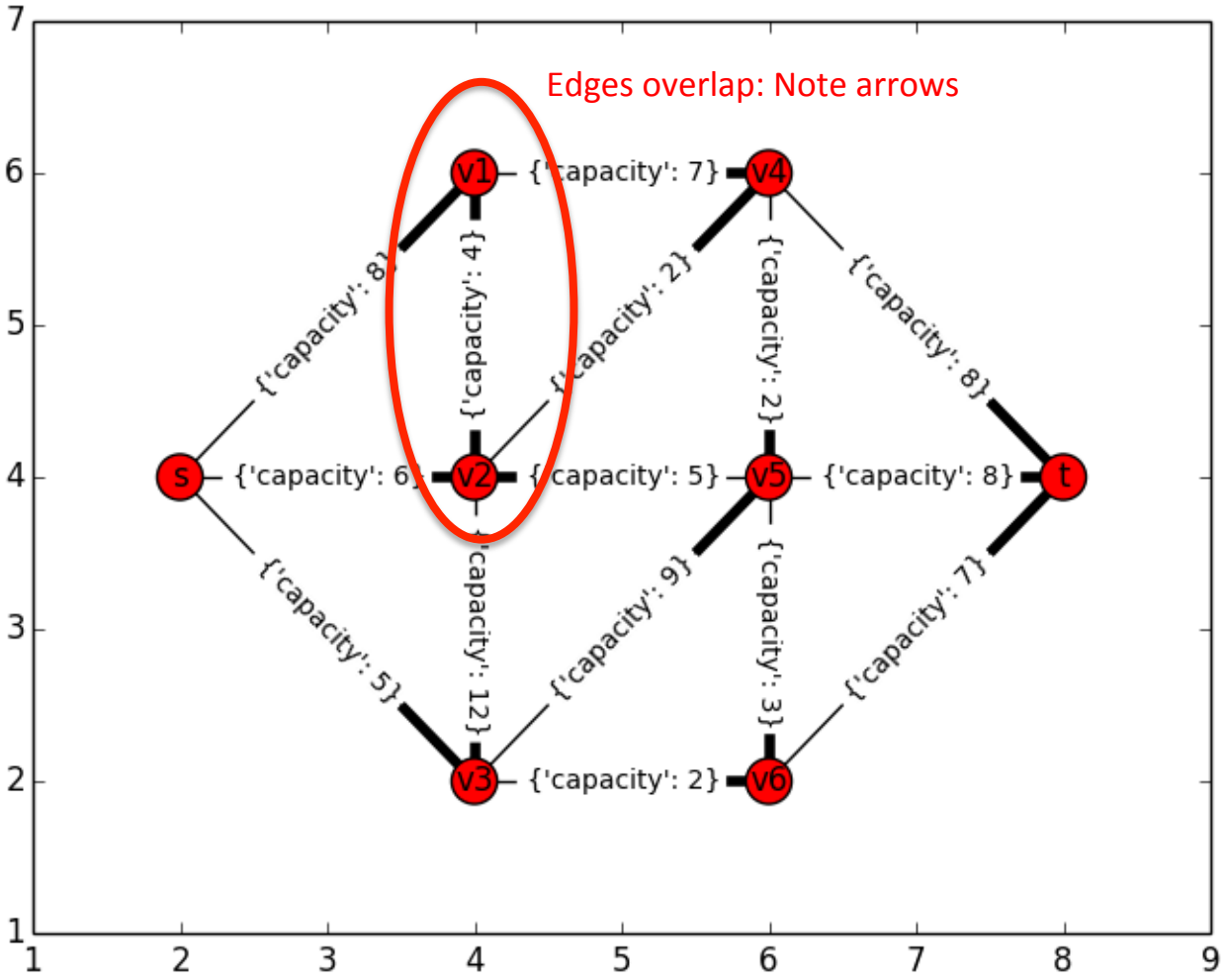


Figure 1: Flow Network

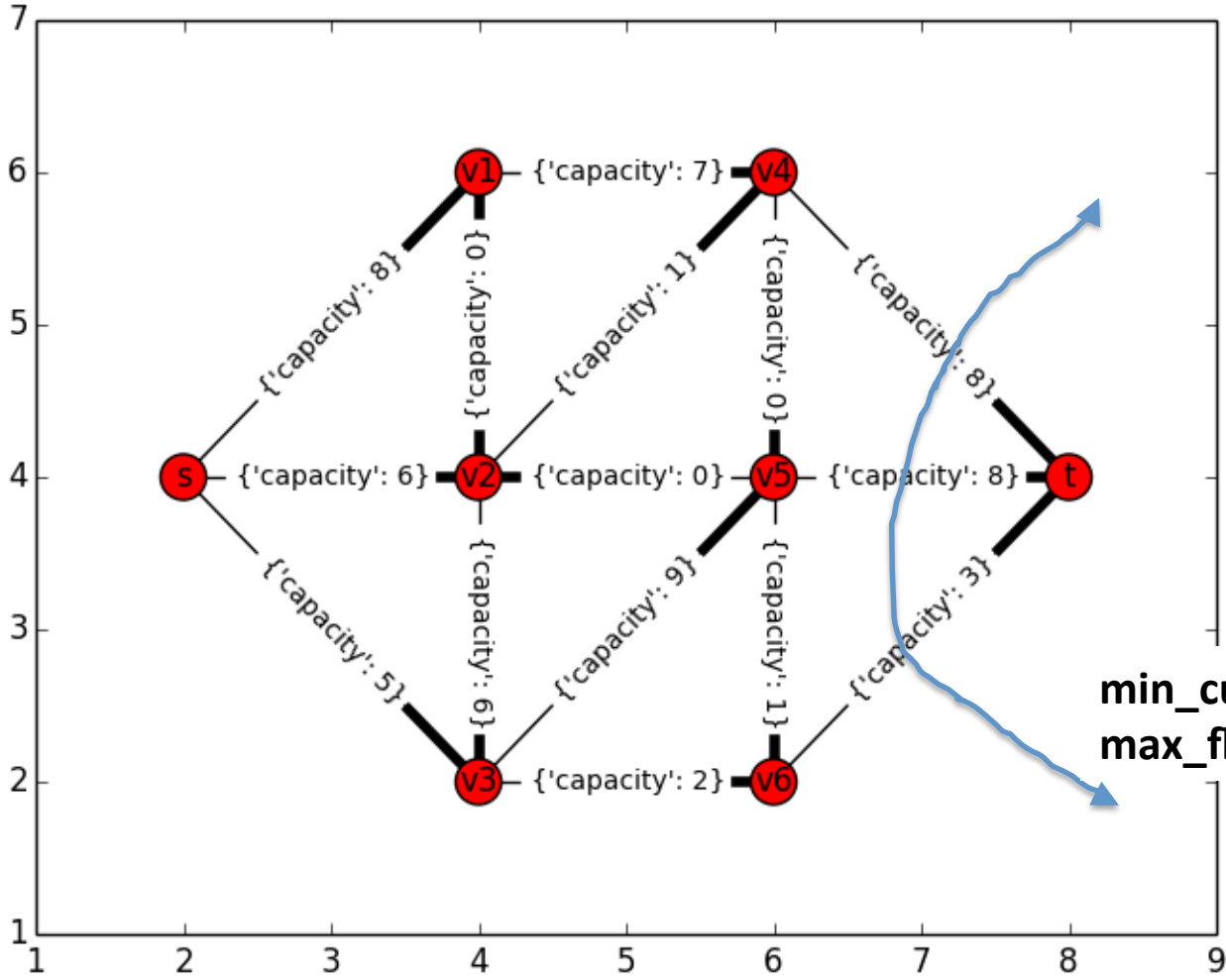


Source:, Analysis of Algorithms, Fall '13, Kouri, USF
References: "Introduction to Algorithms" CLRS, 3rd ed; The Algorithm Design Manual, Skiena

Flow Network Problem Modeled in NetworkX



Resulting Flow Graph (Python / NetworkX)



min_cut: 19
max_flow: 19

Ford_Fulkerson_flow: (Dictinary as source for graph given above)
{'s': {'v1': 8, 'v2': 6, 'v3': 5}, 'v1': {'v2': 1, 'v4': 7}, 'v2': {'v1': 0, 'v3': 6, 'v4': 1}, 'v3': {'v5': 9, 'v6': 2}, 'v4': {'v5': 0, 't': 8}, 'v5': {'v2': 0, 't': 8, 'v6': 1}, 'v6': {'t': 3}, 't': {}}

Exmple Python Code / Execution

```
#
# Jason Maynard
# jlmaynard@mail.usf.edu
#
# Flow network modeling and analysis
#
# Filename:     "network_flow_final_exam.py"
#
# Using NetworkX to model flow networks
# www.networkx.com
# http://networkx.lanl.gov/reference/algorithms.flow
#
# Source:       Final exam, Analysis of Algorithms,
#
# References:   "Introduction to Algorithms", CLRS,
#               "The Algorithm Design Manual", Skien
#
# Notes: See associated PPT file "flow_network_jlm.p
#
# Example output:
# The min cut is:  19
# The max flow is:  19
# Ford_Fulkerson_flow:
# {'s': {'v1': 8, 'v2': 6, 'v3': 5},
# 'v1': {'v2': 1, 'v4': 7},
# 'v2': {'v1': 0, 'v3': 6, 'v4': 1},
# 'v3': {'v5': 9, 'v6': 2},
# 'v4': {'v5': 0, 't': 8},
# 'v5': {'v2': 0, 't': 8,
# 'v6': 1}, 'v6': {'t': 3},
# 't': {}}
#

import networkx as nx                   # Network mo
import matplotlib.pyplot as plt         # Graph plot
import array
```

```
# GRAPH MODEL ------------------------------------------------------------

# Instantiate graph object
G = nx.DiGraph()

# Build graph (Hard coded here but could be changed to read input file)
G.add_edge('s','v1', capacity = 8)
G.add_edge('s','v2', capacity = 6)
G.add_edge('s','v3', capacity = 5)

G.add_edge('v1','v2', capacity = 5)
G.add_edge('v1','v4', capacity = 7)

G.add_edge('v2','v1', capacity = 4)
G.add_edge('v2','v4', capacity = 2)
G.add_edge('v2','v3', capacity = 12)

G.add_edge('v3','v5', capacity = 9)
G.add_edge('v3','v6', capacity = 2)

G.add_edge('v4','v5', capacity = 2)
G.add_edge('v4','t', capacity = 8)

G.add_edge('v5','v2', capacity = 5)
G.add_edge('v5','v6', capacity = 3)
G.add_edge('v5','t', capacity = 8)

G.add_edge('v6','t', capacity = 7)

# PLOT GRAPH G - ( Comment / Uncomment as needed... )
# Position the nodes as desired
pos = {
    's' :(2,4),
    'v1':(4,6),
    'v2':(4,4),
    'v3':(4,2),
    'v4':(6,6),
    'v5':(6,4),
    'v6':(6,2),
    't' :(8,4)
    }

# Draw graph G at the pos described above
nx.draw_networkx(G, pos)

# Add edge labels
nx.draw_networkx_edge_labels(G,pos)

# Display the graph
plt.show()
```

```
# ANALYSIS ---------------------------------------------------------------
min_cut = nx.min_cut(G, 's', 't')
max_flow = nx.max_flow(G, 's', 't')
fff = nx.ford_fulkerson_flow(G, 's', 't')   # returns dict


# RESULTS ----------------------------------------------------------------
print 'min_cut:  ', min_cut
print 'max_flow: ', max_flow
print 'Ford_Fulkerson_flow: \n', fff

# Build resulting flow graph
Gf = nx.DiGraph()

# Generate list of keys (i.e., Nodes)
node = fff.keys()

# "index" into dictionary result generated in Ford_Fulkerson_Flow

Gf.add_edge(node[0], node[1], capacity = fff['s']['v1'])    # s -> v1 = 8
Gf.add_edge(node[0], node[2], capacity = fff['s']['v2'])    # s -> v2 = 6
Gf.add_edge(node[0], node[3], capacity = fff['s']['v3'])    # s -> v3 = 5

Gf.add_edge(node[1], node[2], capacity = fff['v1']['v2'])   # v1 -> v2 = 1
Gf.add_edge(node[1], node[4], capacity = fff['v1']['v4'])   # v1 -> v4 = 7

Gf.add_edge(node[2], node[1], capacity = fff['v2']['v1'])   # v2 -> v1 = 0
Gf.add_edge(node[2], node[3], capacity = fff['v2']['v3'])   # v2 -> v3 = 6
Gf.add_edge(node[2], node[4], capacity = fff['v2']['v4'])   # v2 -> v4 = 1

Gf.add_edge(node[3], node[5], capacity = fff['v3']['v5'])   # v3 -> v5 = 9
Gf.add_edge(node[3], node[6], capacity = fff['v3']['v6'])   # v3 -> v6 = 2

Gf.add_edge(node[4], node[5], capacity = fff['v4']['v5'])   # v4 -> v5 = 0
Gf.add_edge(node[4], node[7], capacity = fff['v4']['t'])    # v4 ->  t = 8

Gf.add_edge(node[5], node[2], capacity = fff['v5']['v2'])   # v5 -> v2 = 0
Gf.add_edge(node[5], node[7], capacity = fff['v5']['t'])    # v5 -> t =  8
Gf.add_edge(node[5], node[6], capacity = fff['v5']['v6'])   # v5 -> v6 = 1

Gf.add_edge(node[6], node[7], capacity = fff['v6']['t'])    # v6 -> t =  3


# Draw graph Gf at the pos described above
nx.draw_networkx(Gf, pos)

# Add edge labels
nx.draw_networkx_edge_labels(Gf,pos)

# Display the graph
plt.show()
```

```
>>> ================================ RESTART ================================
>>>
min_cut:  19
max_flow:  19
Ford_Fulkerson_flow:
{'s': {'v1': 8, 'v2': 6, 'v3': 5}, 'v1': {'v2': 1, 'v4': 7}, 'v2': {'v1': 0, 'v3': 6, '
v4': 1}, 'v3': {'v5': 9, 'v6': 2}, 'v4': {'v5': 0, 't': 8}, 'v5': {'v2': 0, 't': 8, 'v6
': 1}, 'v6': {'t': 3}, 't': {}}
>>>
```