**ENGN2020 – HOMEWORK5**

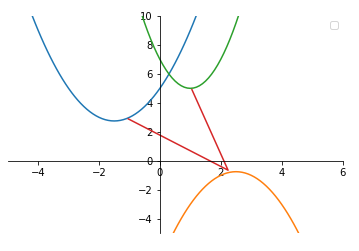
### Problem 1

### (a) formal optimization:

minimize

subject to:

### (c) formal optimization:



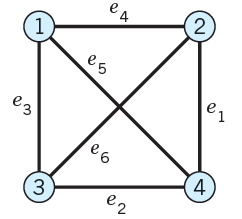
**Fig 1.** Three parabolas and shortest path from p1 to p2 to p3

### Problem 2

### (a) K23-1-10

#### Answer:

The given graph is shown in Fig 1.

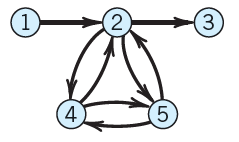


**Fig 2.** The given graph

The adjacency matrix of this graph is:

#### (a) K23-1-12

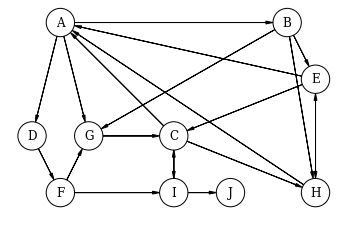
The given digraph is shown in Fig 2.



**Fig 3.** The given digraph

The adjacency matrix of this digraph is:

### Problem 4



**Fig 4.** The given digraph

Code:

The definition of the “*Vertex”* class is shown as below:

from matplotlib import rc

rc("font", family="serif", size=12)

rc("text", usetex=False)

#import daft library

import daft

class Vertex:

#return the edges based on input vertex name

def get\_children(self,name):

self.name = name

return Vertex.links[self.name];

#draw the graph

def draw(self):

#initialize the graph

pgm = daft.PGM([6, 6], origin=[0, 1])

#give the location of all vertices

pgm.add\_node(daft.Node("A", r"A", 1, 5))

pgm.add\_node(daft.Node("B", r"B", 5, 5))

pgm.add\_node(daft.Node("D", r"D", .5, 3))

pgm.add\_node(daft.Node("G", r"G", 1.5, 3))

pgm.add\_node(daft.Node("E", r"E", 5.5, 4))

pgm.add\_node(daft.Node("H", r"H", 5.5, 2))

pgm.add\_node(daft.Node("C", r"C", 3, 3))

pgm.add\_node(daft.Node("F", r"F", 1, 1.5))

pgm.add\_node(daft.Node("I", r"I", 3, 1.5))

pgm.add\_node(daft.Node("J", r"J", 5, 1.5))

#loop all edges to add to daft

for start, edges in Vertex.links.items():

for item in edges:

pgm.add\_edge(start,item)

#draw the graph

pgm.render()

pgm.figure.savefig("vertex.png",dpi=150)

#the graph

links = {

'A': ['B', 'D', 'G'],

'B': ['E', 'G', 'H'],

'C': ['A', 'H', 'I'],

'D': ['F'],

'E': ['H', 'A', 'C'],

'F': ['G', 'I'],

'G': ['C'],

'H': ['A', 'E'],

'I': ['C', 'J']

}

#define a Vertex object

a = Vertex()

#draw the graph

a.draw()