ENGN2020 – HOMEWORK5

Problem 1

(a) formal optimization:

minimize $f(x_1, x_2, x_3, x_4, x_5, x_6) = \sqrt{(x_1 - x_3)^2 + (x_2 - x_4)^2} + \sqrt{(x_3 - x_5)^2 + (x_4 - x_6)^2}$

subject to:

$$g_1(x_1, x_2) = x_1^2 + 3x_1 + 5 - x_2 = 0$$
$$g_2(x_3, x_4) = 2x_3^2 - 4x_3 + 5 - x_4 = 0$$
$$g_3(x_5, x_6) = -x_5^2 + 5x_5 + 5 - x_6 = 0$$

(c) formal optimization:

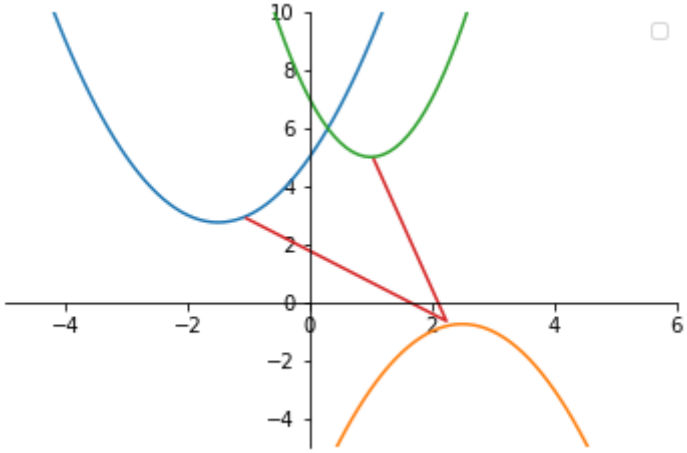

Fig 1. Three parabolas and shortest path from p1 to p2 to p3

Problem 2

(a) K23-1-10

Answer:

The given graph is shown in Fig 1.

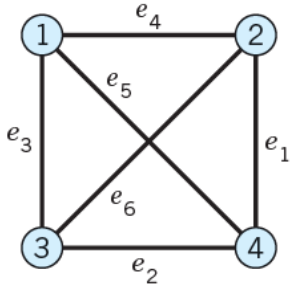

Fig 2. The given graph

The adjacency matrix of this graph is:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

(a) K23-1-12

The given digraph is shown in Fig 2.

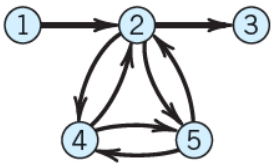

Fig 3. The given digraph

The adjacency matrix of this digraph is:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Problem 4

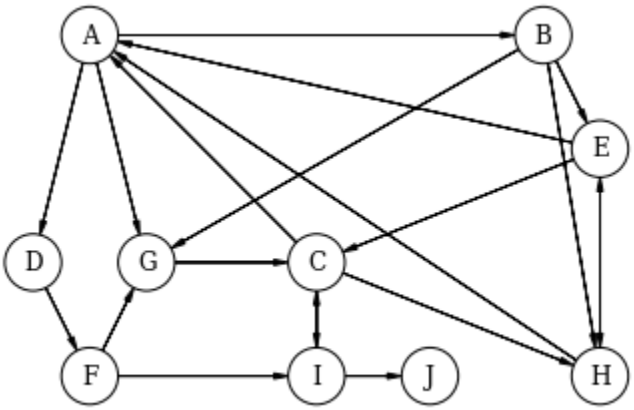

Fig 4. The given digraph

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()
```