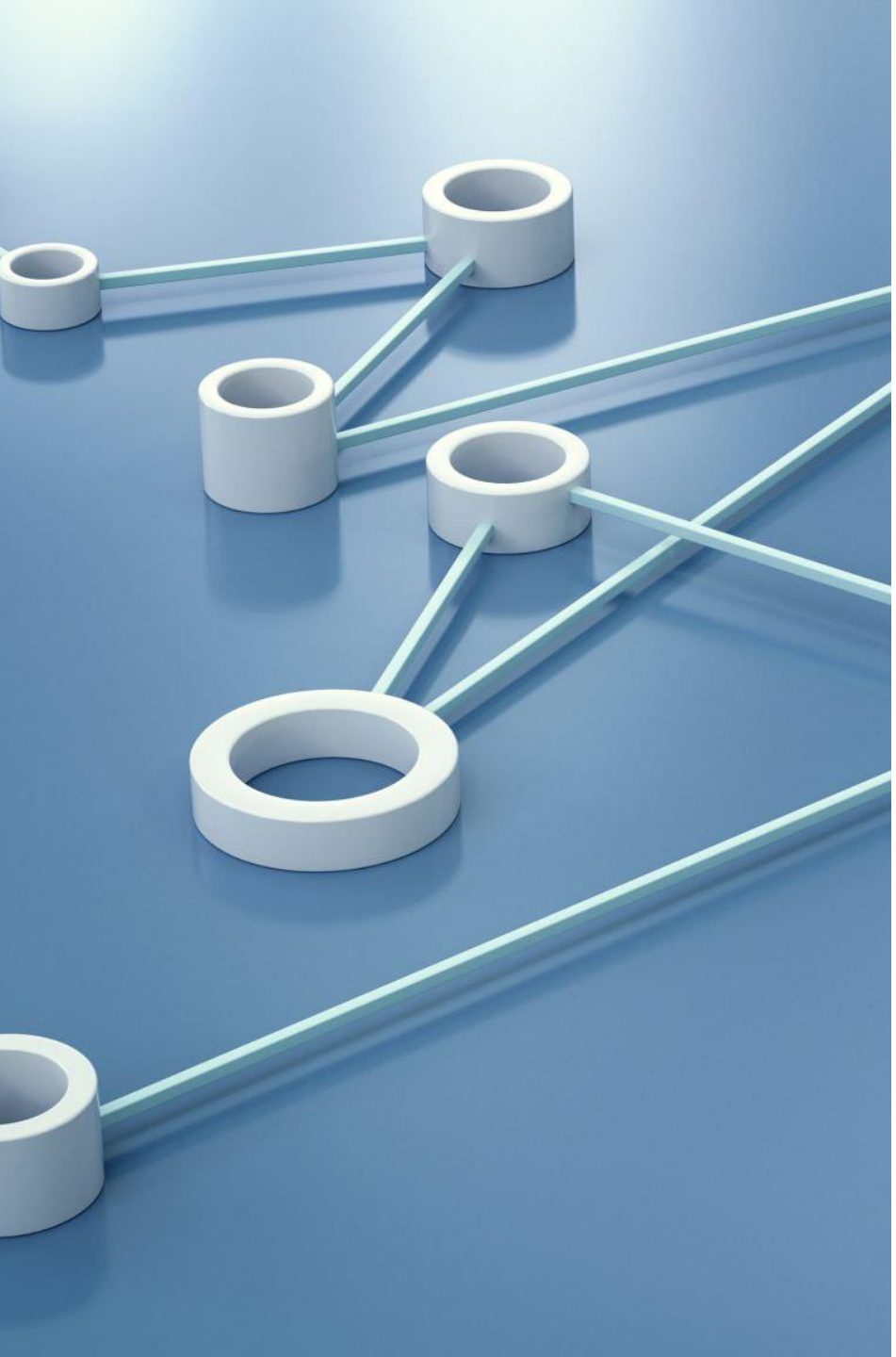
Project proposal

Network Topology

Abhishek M (200070003)
Neeraj Nixon (20D070056)

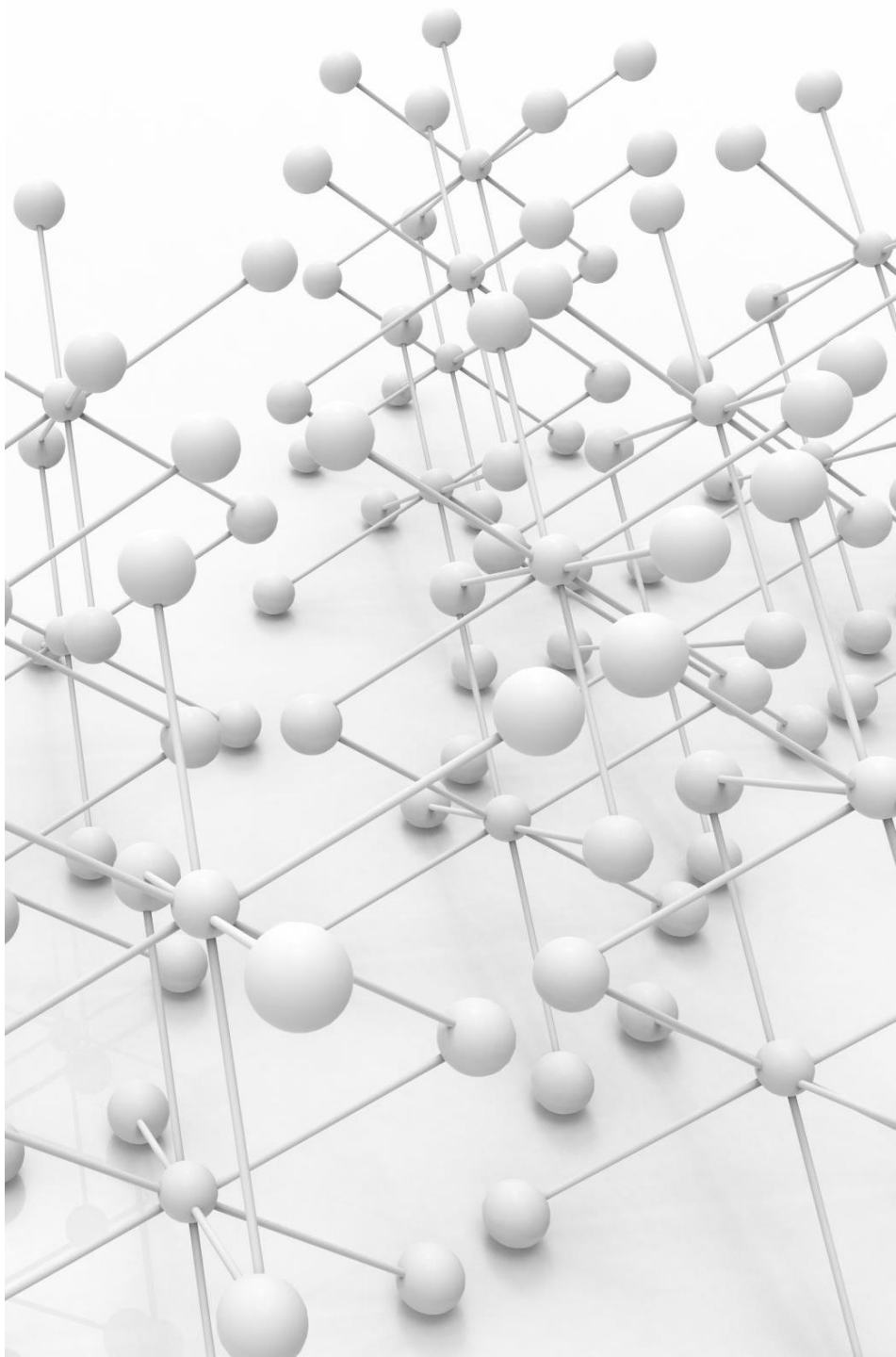


Network Topology

- **Network topology** is the arrangement of the elements of a communication network. Network topology can be used to define or describe the arrangement of various types of telecommunication networks.
- Each node in the network topology has a certain set of parameters associated with it like Latency and Bandwidth.

Aim

- To create a logical program which can create a network topology which minimizes the path between two given nodes.
- To make a python GUI to create an interactive user interface.



Components involved

The clingo code which implements the required topology by connecting the nodes which minimizes the path between them.

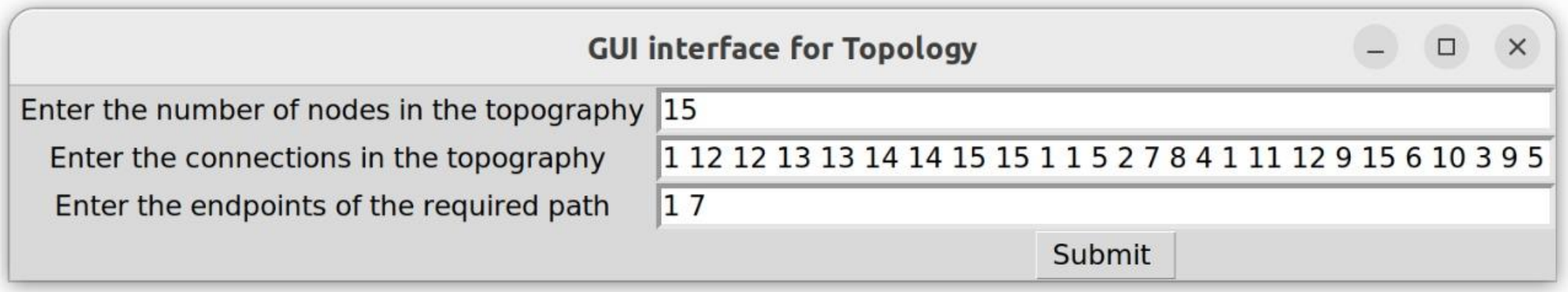
A python code that implements the GUI interface for the program which provides the user with a better way to interact with the code

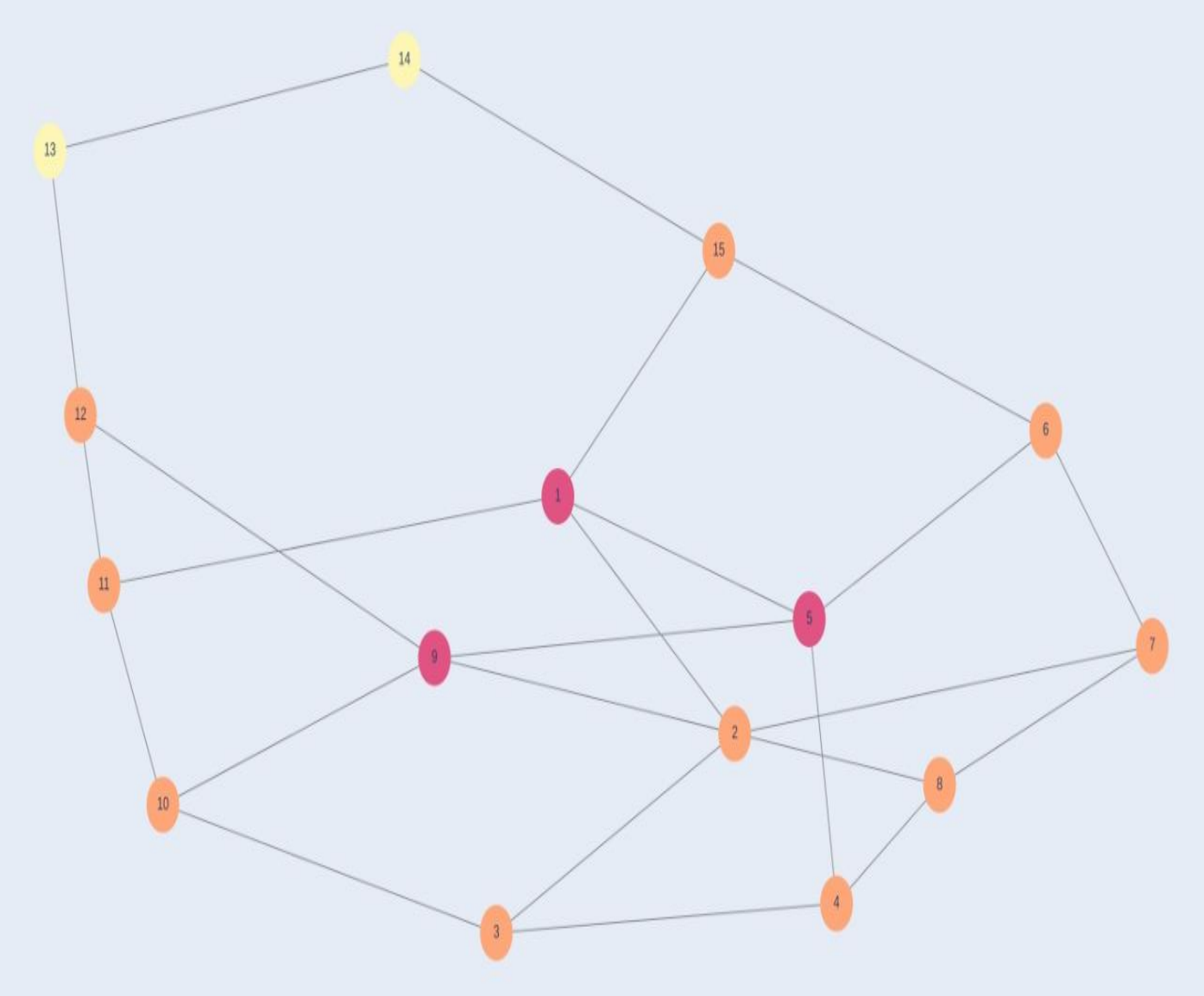
Input format

Number of nodes and end points between which the minimum path is to be calculated within the network topology.

The connection between the nodes within the network topology (user defined).

Python GUI Input Format



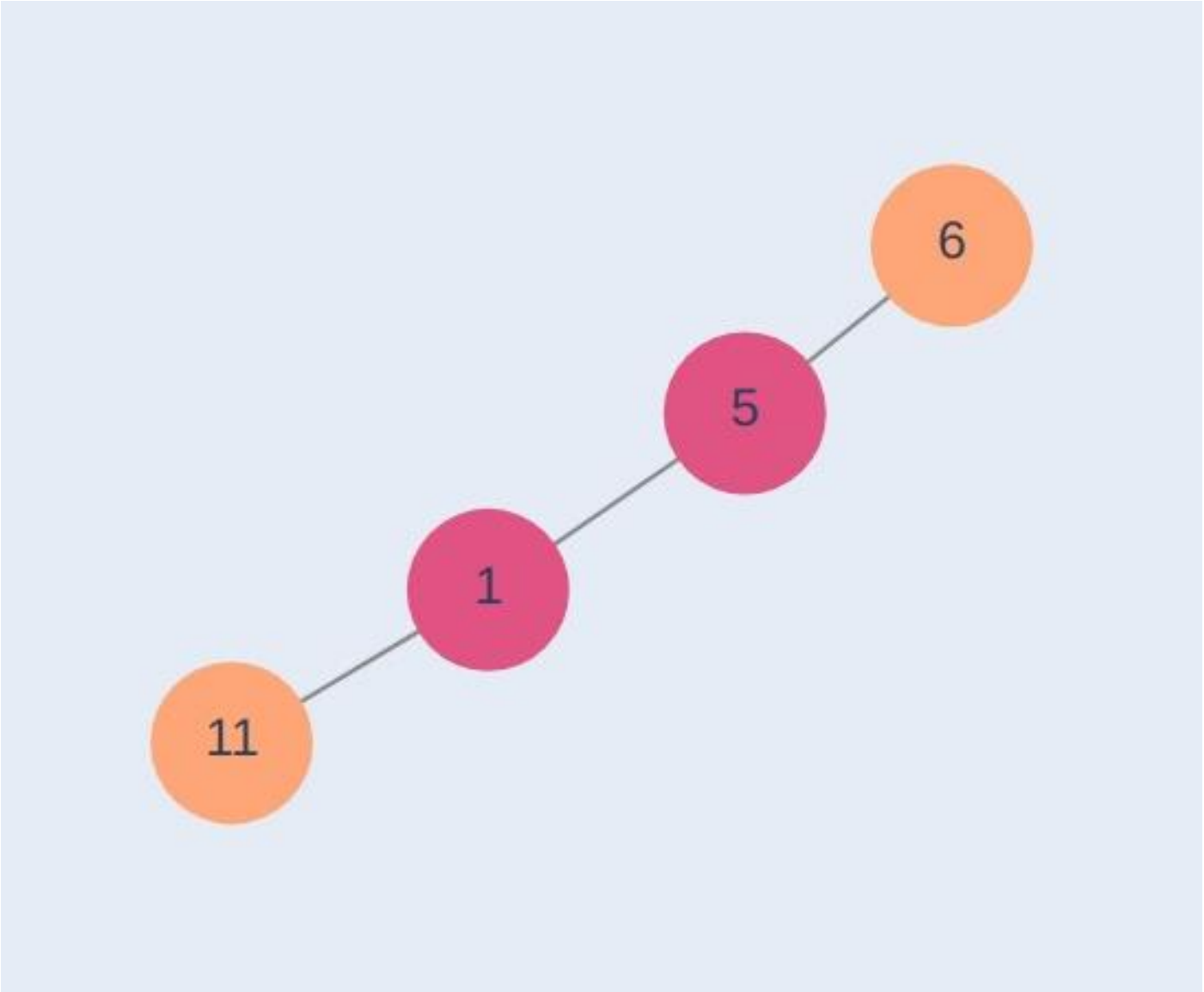


Output format

- Outputs the network topology created by the user with all nodes and connections.

Output (contd.)

- The end output is any one of the minimum paths possible between the two user defined nodes.



Progress

1

We have the clingo code to minimize the path between two given nodes

2

We have completed the input and output GUI interface using python.

3

We have almost completed interfacing clingo with the python GUI.

Clingo Code

```
1
2 %{path(X,Y)}:-node(X,_),node(Y,_),X<Y.
3 %:-#count{X,Y:path(X,Y)}!=1.
4 %path(2,8).
5
6 {nod(X)}:-node(X).
7 :-path(X,Y),not nod(X),nod(Y).
8 :-path(X,Y),nod(X),not nod(Y).
9 :-path(X,Y),not nod(X),not nod(Y).
10 :-path(X,Y), N1=#count{X1:connect(X1,Y), nod(X1)},N2=#count{X1:connect(Y,X1), nod(X1)},N1+N2!=1.
11 :-path(X,Y), N3=#count{X1:connect(X,X1), nod(X1)},N4=#count{X1:connect(X1,X), nod(X1)},N3+N4!=1.
12 :-path(X,Y), X1!=X, X1!=Y, nod(X1), N1=#count{X2:connect(X1,X2),nod(X2)},N2=#count{X2:connect(X2,X1), nod(X2)}, N1+N2!=2.
13
14 %{con(X,Y)}:-connect(X,Y).
15 %:-path(X,Y), N1=#count{X1:con(X1,Y)},N2=#count{X1:con(Y,X1)},N1+N2!=1.
16 %:-path(X,Y), N3=#count{X1:con(X,X1)},N4=#count{X1:con(X1,X)},N3+N4!=1.
17 %:-path(X,Y), N1=#count{X2:con(X1,X2),X1!=X, X1!=Y},N2=#count{X2:con(X2,X1),X1!=X, X1!=Y}, N1+N2!=2.
18
19 %#minimize{1,X,Y:con(X,Y)}.
20
21
22 n(X,Y,N):-path(X,Y), N=#count{X1:nod(X1)}.
23
24 #minimize{1,X1:nod(X1)}.
25
26
27 %n(X,Y,N):-path(X,Y), N=#count{X1:nod(X1)}.
28
29
30 %#minimize{1,X1:nod(X1),connect(X,Y)}.
31
32
33 %#show con/2.
34 #show n/3.
35 %#show  path/2.
36 #show nod/1.
37
38
```

```
node(1,3).
node(2,3).
node(3,3).
node(4,3).
node(5,3).
node(6,3).
node(7,3).
node(8,2).
node(9,3).
node(10,4).



{connect(1..6,1..6)}.
:-connect(X,Y),X>=Y.

%:-N1=#count{X:connect(X,Y)},N2=#count{X:connect(Y,X)},N<N1+N2,node(Y,N).
%:-N1=#count{X:connect(X,Y)},N2=#count{X:connect(Y,X)},2>N1+N2,node(Y,N).
:-N1=#count{X:connect(X,Y)},N2=#count{X:connect(Y,X)},N1+N2!=N,node(Y,N).
%#maximize{1,X,Y:connect(X,Y)}.

#show connect/2.

```

Integrating Clingo with Python

```python

# Integrating Clingo

import clingo

def answer(ans):
    # Creating the from and to lists

    from_list2 = []
    to_list2 = []
    newnode = []

    atoms=ans.symbols(atoms=True)
    for atom in atoms:
        if str(atom.name)=="nod":
            newnode.append(atom.arguments[0].number)
        if str(atom.name)=="n":
            num_nodes=atom.arguments[2].number

    print(num_nodes)
    print(newnode)
```

```python
#-----------------------------------------------------------

ctl=clingo.Control("0")
with open("data1.lp",'a') as f:
    for i in node_list:
        f.write("node("+i+").")
    for j in range(0,len(from_list)):
        f.write("connect("+from_list[j]+","+to_list[j]+").")
   # f.write("path("+pathends[0]+","+pathends[1]+").")
ctl.load("data1.lp")
ctl.load("Project.lp")
ctl.configuration.solve.models="1"
ctl.ground([("base",[])])

with ctl.solve(on_model=lambda m: answer(m),async_=True) as handle:
    while not handle.wait(0):pass
    handle.get()
```

Progress After last Presentation

Clingo code for finding the smallest path between any two nodes in a network were written. Here the clingo code has been now modified to find the least weighted path between any two user defined nodes in the network topology (The weights for each edge /connection is user defined.

Integrating Python GUI with Clingo was not done last time. Now integrating Clingo with python has been implemented successfully .

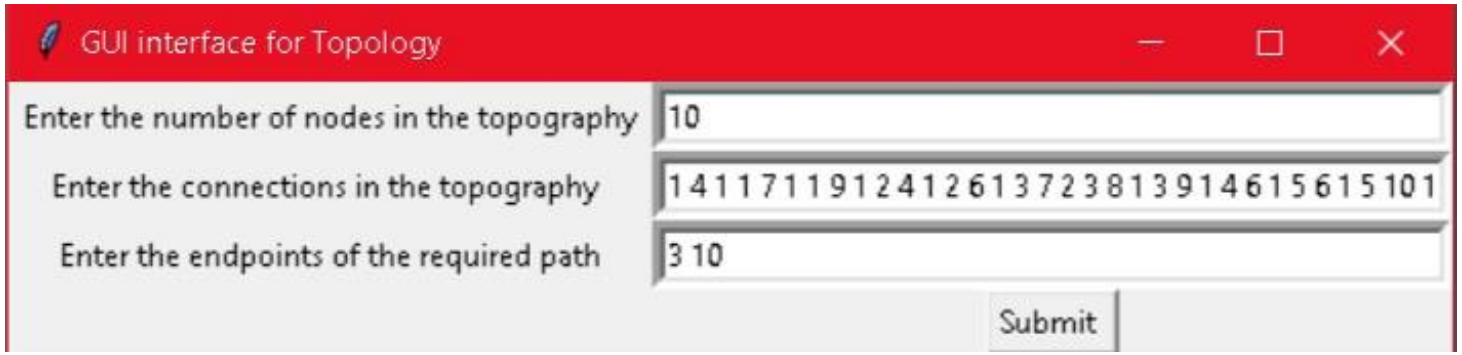
Clingo Code

clin.lp

```
{nod(X)}:-node(X).

:-path(X,Y),not nod(X),nod(Y).

:-path(X,Y),nod(X),not nod(Y).

:-path(X,Y),not nod(X),not nod(Y).

:-path(X,Y),N=#count{X1:nod(X1),X1!=X, X1!=Y},N=0, N1=#count{X1:connect(X1,Y,_), nod(X1)},N2=#count{X1:connect(Y,X1,_), nod(X1)},N1+N2!=1.

:-path(X,Y),N=#count{X1:nod(X1),X1!=X, X1!=Y},N=0, N3=#count{X1:connect(X,X1,_), nod(X1)},N4=#count{X1:connect(X1,X,_), nod(X1)},N3+N4!=1.

:-path(X,Y), X1!=X, X1!=Y, nod(X1), N1=#count{X2:connect(X1,X2,_),nod(X2)},N2=#count{X2:connect(X2,X1,_), nod(X2)}, N1+N2!=2.

:-path(X,Y),N=#count{X1:nod(X1),X1!=X, X1!=Y},N!=0, N1=#count{X1:connect(X,X1,_),path(X,Y), nod(X1),X1!=X, X1!=Y},
 N2=#count{X1:connect(X1,X,_), nod(X1),path(X,Y), nod(X1),X1!=X, X1!=Y}, N1+N2!=1.

:-path(X,Y),N=#count{X1:nod(X1),X1!=X, X1!=Y},N!=0, N1=#count{X1:connect(Y,X1,_),path(X,Y), nod(X1),X1!=X, X1!=Y},
 N2=#count{X1:connect(X1,Y,_), nod(X1),path(X,Y), nod(X1),X1!=X, X1!=Y}, N1+N2!=1.


1{con(X1,Y1,N)}1:-connect(X1,Y1,N),nod(X1),nod(Y1).

n(X,Y,N):-path(X,Y), N=#sum{N10,X1,Y1:con(X1,Y1,N10)}, 1=#count{N10,X1,Y1:con(X1,Y1,N10)}.

n(X,Y,N):-path(X,Y), N1=#sum{N10,X1,Y1:con(X1,Y1,N10)}, N2=#sum{N20,X1,Y1:con(X1,Y1,N20), path(X,Y), X1=X, Y1=Y},
N3=#sum{N30,X1,Y1:con(X1,Y1,N30), path(X,Y), X1=Y, Y1=X}, N=N1-N2-N3,1<#count{N10,X1,Y1:con(X1,Y1,N10)}.

#minimize{N:n(X1,Y1,N)}.

#show con/3.
#show n/3.
#show nod/1.
```

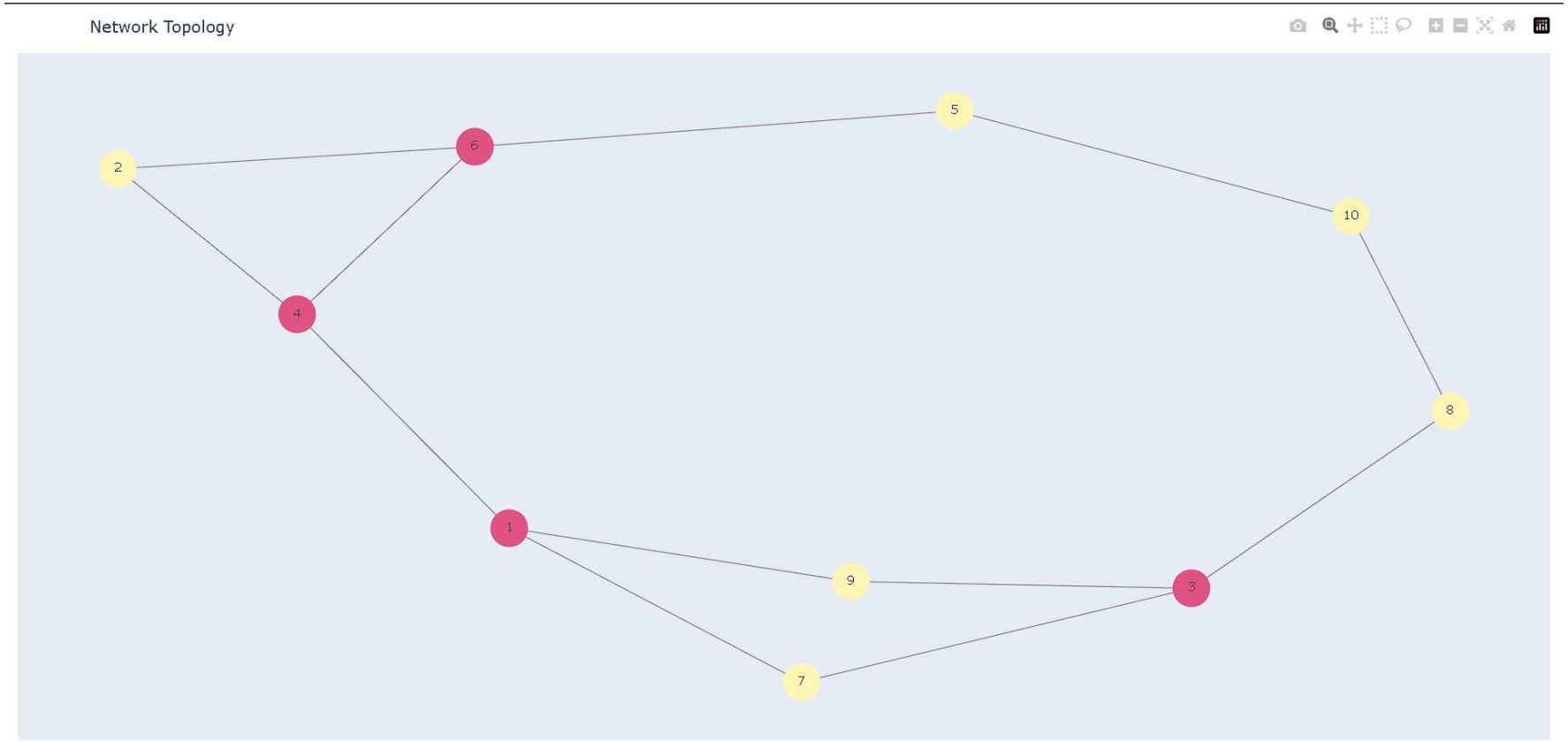
Sample Input

- The first box specifies number of nodes in the network.
- The second box takes each connections and their weights
- (eg:1 4 1 1 7 1 .... means nodes 1 and 4 connected with weight 1, nodes1 and 7 connected with weights 1 and so on)
- The third box takes the nodes between which you need to find the least weighted path.



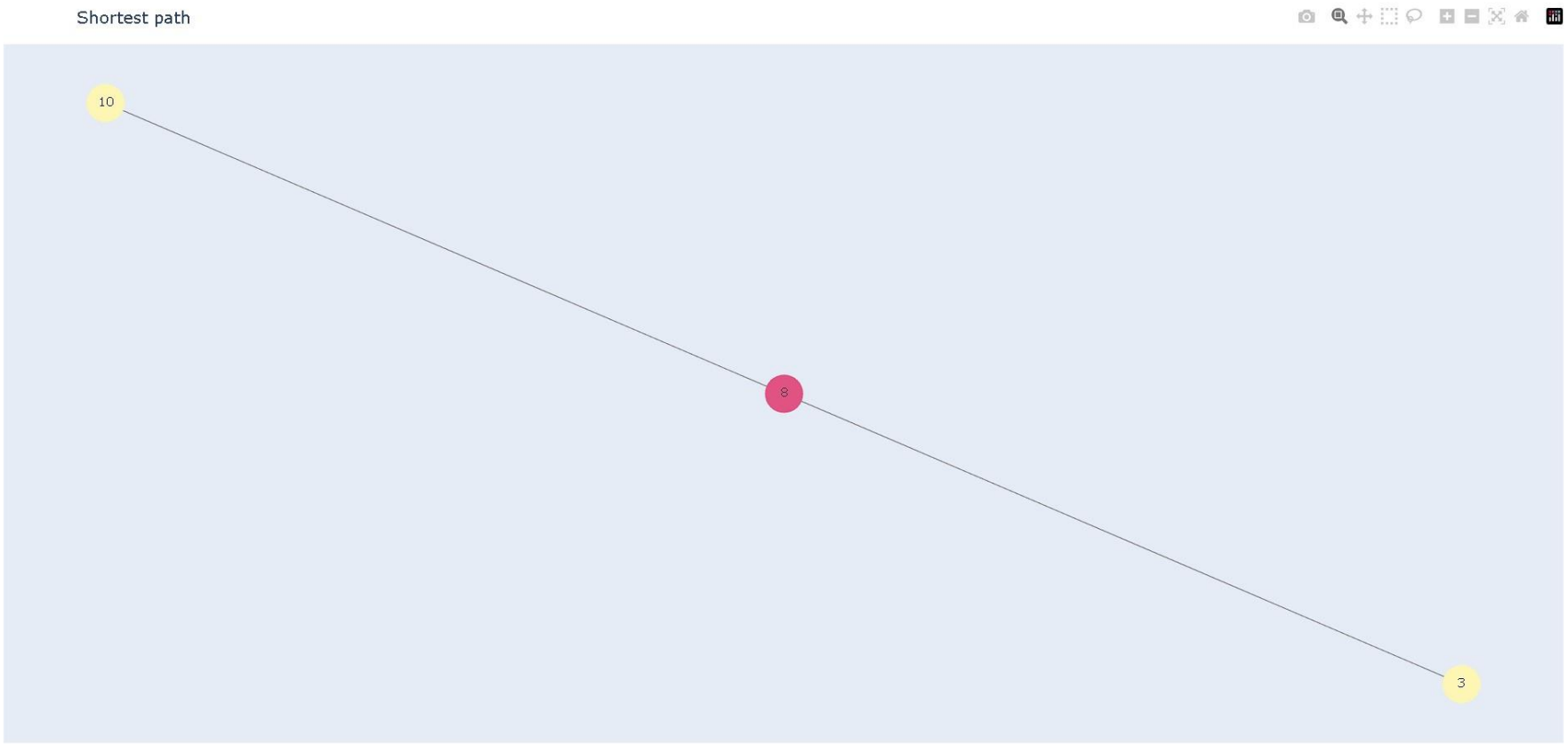
Sample Output

Outputs the network topology created by the user with all nodes and connections.



Output(contd.)

The end output is the least weighing path possible between the two user defined nodes.



Thank You