' Copyrights 2016 Ahmad F. Al Musawi

Public Class Form1

Dim myg As Graphics

Const N = 20 'number of nodes

Dim A As Integer = 6

Dim L\_max, L\_min As Integer

Public Structure position

Dim x As Integer

Dim y As Integer

End Structure

Public Structure edges

Public attached As Boolean

Public ID1\_indextonodes As Integer

Public ID2\_indextonodes As Integer

End Structure

Dim Text As String

Dim Simulating As Boolean

Dim NumberofSimulation As Integer

Dim motifList(1000000) As Motif

Dim iMotif As Integer

Dim V(N) As vertex ' set of vertices

Dim network(N, N) As Integer ' adjancecy matrix

Dim area, k As Integer

Dim H, W As Integer

Dim Goal\_NW\_Nodes(1666) As vertex : Dim i4Goal\_NW\_Nodes As Integer

Dim GoalNetworkAdjancyMatrix(1566, 1566) As Boolean

Dim GoalNetworkAdjancyMatrix\_checkCorrection(1566, 1566) As Integer

Dim i4edges2 As Integer

Dim GoalNetwork(1566) As edges

Dim changing As Integer

Dim e1, e2 As Integer

Dim newnet As Integer

Dim xm, ym, Ti As Integer ' Ti is the selected node!

Dim ee As Integer

Dim clicked As Boolean = False

Dim Motif As Integer

Dim iM, jM, kM As Integer

Dim MoveNode As Boolean

Dim delete\_List(1666) As vertex : Dim iDelete As Integer

Private Sub define\_new\_motif(ByVal i As Integer, ByVal j As Integer, ByVal k As Integer)

iMotif = iMotif + 1

With motifList(iMotif)

.Node1 = Goal\_NW\_Nodes(i)

.Node2 = Goal\_NW\_Nodes(j)

.node3 = Goal\_NW\_Nodes(k)

'-------------------------------

ReDim .e(3, 3)

'preventing cycle in motif

.e(1, 1) = False 'GoalNetworkAdjancyMatrix(i, i)

.e(1, 2) = GoalNetworkAdjancyMatrix(i, j)

.e(1, 3) = GoalNetworkAdjancyMatrix(i, k)

'------------------------------------------

.e(2, 1) = GoalNetworkAdjancyMatrix(j, i)

.e(2, 2) = False 'GoalNetworkAdjancyMatrix(j, j)

.e(2, 3) = GoalNetworkAdjancyMatrix(j, k)

'------------------------------------------

.e(3, 1) = GoalNetworkAdjancyMatrix(k, i)

.e(3, 2) = GoalNetworkAdjancyMatrix(k, j)

.e(3, 3) = False 'GoalNetworkAdjancyMatrix(k, k)

End With

If (Motif\_Type(motifList(iMotif), 1)) Then

ListBox1.Items.Add(j & "<-- " & i & " -->" & k)

End If

End Sub

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

H = PictureBox1.Height

W = PictureBox1.Width

area = H \* W

k = Math.Sqrt(area / N)

myg = PictureBox1.CreateGraphics

For i = 1 To N

With V(i)

.ID = i

.X = (W / 2) + (changing \* (Rnd() \* W / 2))

.Y = (H / 2) + (changing \* (Rnd() \* H / 2))

End With

Next

L\_max = HScrollBar1.Value

L\_min = HScrollBar2.Value

End Sub

Public Function distance(ByVal i As Integer, ByVal j As Integer) As Integer

Dim x1 As Integer = Goal\_NW\_Nodes(i).X

Dim y1 As Integer = Goal\_NW\_Nodes(i).Y

Dim x2 As Integer = Goal\_NW\_Nodes(j).X

Dim y2 As Integer = Goal\_NW\_Nodes(j).Y

Dim x, y As Integer

x = x1 - x2

y = y1 - y2

Dim z = Math.Sqrt((x \* x) + (y \* y))

Return z

End Function

Private Sub circle(ByVal X As Integer, ByVal Y As Integer, ByVal R As Integer, Optional ByVal colorr As Integer = 1)

Select Case colorr

Case 1

Dim clsBrush As New System.Drawing.SolidBrush(Color.Blue)

Case 2

End Select

'myg.FillEllipse(clsBrush, X - 5, Y - 5, 15, 15)

Dim xn, yn As Integer

xn = X - (0.5 \* R)

yn = Y - (0.5 \* R)

myg.DrawEllipse(Pens.Black, xn, yn, R, R)

End Sub

Private Sub selected\_circle(ByVal X As Integer, ByVal Y As Integer, ByVal name As Integer)

Dim clsBrush As New System.Drawing.SolidBrush(Color.Red)

myg.FillEllipse(clsBrush, X - 10, Y - 10, 20, 20)

With TextBox2

.Text = name & vbNewLine

.Text = .Text & "Indegree: " & in\_Degrees(Index(Goal\_NW\_Nodes(name))) & vbNewLine

.Text = .Text & "Outdegree: " & out\_Degrees(Index(Goal\_NW\_Nodes(name))) & vbNewLine

End With

End Sub

Private Sub check\_limit(ByVal i As Integer, ByVal j As Integer)

constrain(Goal\_NW\_Nodes(i).X, W)

constrain(Goal\_NW\_Nodes(i).Y, H)

constrain(Goal\_NW\_Nodes(j).X, W)

constrain(Goal\_NW\_Nodes(j).Y, H)

End Sub

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

For i = 1 To i4Goal\_NW\_Nodes

For j = 1 To i4Goal\_NW\_Nodes

If i <> j Then

If GoalNetworkAdjancyMatrix(i, j) Then

'connected

Dim d As Integer = distance(i, j)

If d > L\_max Then

While d > L\_max 'attract

If out\_Degrees(Index(Goal\_NW\_Nodes(i))) >= out\_Degrees(Index(Goal\_NW\_Nodes(j))) Then

'If Goal\_NW\_Nodes(j).In\_degree = 1 Then L\_max = 50

ATTRACT(Goal\_NW\_Nodes(i).X, Goal\_NW\_Nodes(i).Y, Goal\_NW\_Nodes(j).X, Goal\_NW\_Nodes(j).Y)

Else

'If Goal\_NW\_Nodes(i).In\_degree = 1 Then L\_max = 50

ATTRACT(Goal\_NW\_Nodes(j).X, Goal\_NW\_Nodes(j).Y, Goal\_NW\_Nodes(i).X, Goal\_NW\_Nodes(i).Y)

End If

check\_limit(i, j)

'drawing()

d = distance(i, j)

End While

Else

While d < L\_min 'repulsive

If out\_Degrees(Index(Goal\_NW\_Nodes(i))) >= out\_Degrees(Index(Goal\_NW\_Nodes(j))) Then

repulsive(Goal\_NW\_Nodes(i).X, Goal\_NW\_Nodes(i).Y, Goal\_NW\_Nodes(j).X, Goal\_NW\_Nodes(j).Y)

Else

repulsive(Goal\_NW\_Nodes(j).X, Goal\_NW\_Nodes(j).Y, Goal\_NW\_Nodes(i).X, Goal\_NW\_Nodes(i).Y)

End If

check\_limit(i, j)

'drawing()

d = distance(i, j)

End While

End If

Else

'not connected

Dim d As Integer = distance(i, j)

While d < L\_min 'repulsive

If out\_Degrees(Index(Goal\_NW\_Nodes(i))) >= out\_Degrees(Index(Goal\_NW\_Nodes(j))) Then

repulsive(Goal\_NW\_Nodes(i).X, Goal\_NW\_Nodes(i).Y, Goal\_NW\_Nodes(j).X, Goal\_NW\_Nodes(j).Y)

Else

repulsive(Goal\_NW\_Nodes(j).X, Goal\_NW\_Nodes(j).Y, Goal\_NW\_Nodes(i).X, Goal\_NW\_Nodes(i).Y)

End If

check\_limit(i, j)

'drawing()

d = distance(i, j)

End While

End If

End If

Next

Next

drawing()

End Sub

Private Sub Highlight(ByVal v As vertex)

circle(v.X, v.Y, 30)

End Sub

Private Sub drawing()

myg.Clear(Me.BackColor)

For i = 1 To i4Goal\_NW\_Nodes

For j = 1 To i4Goal\_NW\_Nodes

If GoalNetworkAdjancyMatrix(i, j) Then

Dim x1, y1, x2, y2 As Integer

x1 = Goal\_NW\_Nodes(i).X

y1 = Goal\_NW\_Nodes(i).Y

x2 = Goal\_NW\_Nodes(j).X

y2 = Goal\_NW\_Nodes(j).Y

Dim x3 As Integer = x2 - (15 \* Math.Cos(getangle(x1, x2, y1, y2)))

Dim y3 As Integer = y2 + (15 \* Math.Sin(getangle(x1, x2, y1, y2)))

myg.DrawLine(Pens.Black, x1, y1, x3, y3)

End If

Next

Dim f As Font = New Font("Arial", 12)

Dim tt As String = i '= Goal\_NW\_Nodes(i).In\_degree & ":" & Goal\_NW\_Nodes(i).Out\_degree

myg.DrawString(tt, f, Brushes.Black, Goal\_NW\_Nodes(i).X - 10, Goal\_NW\_Nodes(i).Y - 10)

Next

Dim d = 1000

Dim tx, ty As Integer

For i = 1 To i4Goal\_NW\_Nodes

circle(Goal\_NW\_Nodes(i).X, Goal\_NW\_Nodes(i).Y, 30)

Dim aa As Integer = xm - Goal\_NW\_Nodes(i).X

Dim bb As Integer = ym - Goal\_NW\_Nodes(i).Y

If Math.Sqrt((aa \* aa) + (bb \* bb)) < d Then

d = Math.Sqrt((aa \* aa) + (bb \* bb))

tx = Goal\_NW\_Nodes(i).X

ty = Goal\_NW\_Nodes(i).Y

Ti = i

End If

Next

If Not clicked Then

selected\_circle(tx, ty, Ti)

'If distance(xm, ym, tx, ty) < 50 Then myg.DrawLine(Pens.Red, xm, ym, tx + 5, ty + 5)

Else

If MoveNode Then

Goal\_NW\_Nodes(Ti).X = xm

Goal\_NW\_Nodes(Ti).Y = ym

End If

End If

End Sub

Private Function ReadFile(ByVal F As String) As String

Return My.Computer.FileSystem.ReadAllText(F)

End Function

Private Sub LoadingStaticGraph(ByVal i4File As String)

Dim f As String

On Error GoTo 12

f = ReadFile(i4File)

getting\_GRN\_adjancymatrix(f)

ToolStripStatusLabel1.Text = i4File

Dim r As Integer = 250

Dim xc, yc As Integer

xc = W / 2

yc = H / 2

k = 0

For th = 0 To 360 Step 360 / i4Goal\_NW\_Nodes

k = k + 1

Goal\_NW\_Nodes(k).X = xc + r \* Math.Cos(th \* Math.PI / 180)

Goal\_NW\_Nodes(k).Y = yc - r \* Math.Sin(th \* Math.PI / 180)

Next

RemoveUnconnectedNodes()

Label2.Text = "Edges : " & TotalEdges()

If Not Simulating Then drawing()

Timer1.Enabled = False

Exit Sub

12:

If Not Simulating Then MsgBox("Error in loading the graph." & vbCrLf & i4File)

End Sub

Private Sub getting\_GRN\_adjancymatrix(ByVal strGRN As String)

' this is for processing the text file of the grn network.

Dim x As String

Dim i As Integer = 1

Dim num As Integer

Dim id1, id2 As vertex

'Dim n1, n2 As String

x = Mid(strGRN, 1, 1)

While x <> ";"

i += 1

x = Mid(strGRN, i, 1)

End While

i += 1

strGRN = Mid(strGRN, i + 1, strGRN.Length - i + 1)

For j = 1 To N

x = GetThisQuote(strGRN)

'do the processing here

num = numberOfQoutes(x)

Select Case num

Case Is = 2

Dim v As vertex

v = getBetweenQuotes(x, True)

addNewNode2(v.String\_ID, v.Numeric\_ID, v.Node\_Type)

Case Is = 4

Dim y As vertex

y = getBetweenQuotes(x, True)

addNewNode2(y.String\_ID, y.Numeric\_ID, y.Node\_Type)

'End If

End Select

strGRN = Mid(strGRN, x.Length + 1, strGRN.Length - x.Length + 1)

Next

'over New------------------------------------

While strGRN.Length <> 0

x = GetThisQuote(strGRN)

'do the processing here

num = numberOfQoutes(x)

Select Case num

Case Is = 2

Dim v As vertex

v = getBetweenQuotes(x, True)

addNewNode2(v.String\_ID, v.Numeric\_ID, v.Node\_Type)

Case Is = 4

Dim y As vertex

y = getBetweenQuotes(x, True)

addNewNode2(y.String\_ID, y.Numeric\_ID, y.Node\_Type)

'End If

Case Is = 6

id1 = getBetweenQuotes(x, False)

id2 = getBetweenQuotes(Mid(x, id1.String\_ID.Length + 7, x.Length - (id1.String\_ID.Length + 7)), False)

AddEdge(id1, id2)

End Select

strGRN = Mid(strGRN, x.Length + 1, strGRN.Length - x.Length + 1)

End While

End Sub

Private Function GetThisQuote(ByVal str As String) As String

'delete the first part untill the ;

Dim i As Integer = 1

Dim j As String = ""

Dim x As String = Mid(str, 1, 1)

While x <> ";" And i < str.Length

j = j + x

i += 1

x = Mid(str, i, 1)

End While

Return j + x

End Function

Private Function numberOfQoutes(ByVal str As String) As Integer

Dim x As String

Dim k As Integer = 0

Dim i As Integer = 0

While str.Length <> i

x = Mid(str, i + 1, 1)

If x = Chr(34) Then

k += 1

End If

i += 1

End While

Return k

End Function

Private Function getBetweenQuotes(ByVal str As String, ByVal searchType As Boolean) As vertex

Dim i As Integer = 1

Dim j As New vertex

Dim x As String = Mid(str, i, 1)

While x <> Chr(34)

i += 1

x = Mid(str, i, 1)

End While

i += 1

x = Mid(str, i, 1)

While x <> Chr(34)

j.String\_ID = j.String\_ID + x

i += 1

x = Mid(str, i, 1)

End While

Dim found As Boolean = False

For i = 1 To i4Goal\_NW\_Nodes

If j.String\_ID = Goal\_NW\_Nodes(i).String\_ID Then

found = True

Return Goal\_NW\_Nodes(i)

End If

Next

If Not found Then Return j

End Function

Private Sub addNewNode2(ByVal strID As String, ByVal numID As Integer, ByVal tg As Boolean)

changing \*= -1

i4Goal\_NW\_Nodes += 1

With Goal\_NW\_Nodes(i4Goal\_NW\_Nodes)

.ID = i4Goal\_NW\_Nodes

.String\_ID = strID

.Numeric\_ID = numID

.Node\_Type = tg

.X = W / 4 + Rnd() \* changing \* W / 2

.Y = H / 4 + Rnd() \* changing \* H / 2

End With

End Sub

Private Sub AddEdge(ByRef V1 As vertex, ByRef V2 As vertex)

GoalNetworkAdjancyMatrix(V1.ID, V2.ID) = True

End Sub

Private Function Index(ByVal V As vertex) As Integer

For j = 1 To i4Goal\_NW\_Nodes

If Equals(V, Goal\_NW\_Nodes(j)) Then

Return j

End If

Next

End Function

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

If Timer1.Enabled Then

Timer1.Enabled = False

Button2.Text = "start"

Else

Timer1.Enabled = True

Button2.Text = "Stop"

End If

End Sub

Private Sub OpenToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles OpenToolStripMenuItem.Click

OFD1.ShowDialog()

LoadingStaticGraph(OFD1.FileName)

Goal\_NW\_Nodes = RefineGoalNetwork()

Simulating = False

End Sub

Private Sub LoadNetwork(ByVal S As String)

resetGraph()

LoadingStaticGraph(S)

End Sub

Private Sub ConvertToUndirected()

For i = 1 To i4Goal\_NW\_Nodes

For j = 1 To i4Goal\_NW\_Nodes

If GoalNetworkAdjancyMatrix(i, j) Then GoalNetworkAdjancyMatrix(j, i) = True

If GoalNetworkAdjancyMatrix(j, i) Then GoalNetworkAdjancyMatrix(i, j) = True

Next

Next

End Sub

Private Sub resetGraph()

Timer1.Enabled = False

i4Goal\_NW\_Nodes = 0

i4edges2 = 0

iMotif = 0

changing = 1

newnet = 0

For i = 0 To 1566

For j = 0 To 1566

GoalNetworkAdjancyMatrix(i, j) = False

Next

Next

End Sub

Private Sub Create\_motifs()

'Now after finishing loading the network, we create the motifs existed in it.

For i = 1 To i4Goal\_NW\_Nodes

For j = 1 To i4Goal\_NW\_Nodes

For q = 1 To i4Goal\_NW\_Nodes

If i <> j And j <> k And i <> k Then

define\_new\_motif(i, j, q)

End If

Next

Next

Next

'-------------------------------------------------------------

MsgBox((i4Goal\_NW\_Nodes \* i4Goal\_NW\_Nodes \* i4Goal\_NW\_Nodes) - i4Goal\_NW\_Nodes & " : " & iMotif)

End Sub

Private Sub MassTheGraphToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles MassTheGraphToolStripMenuItem.Click

For i = 1 To i4Goal\_NW\_Nodes

Goal\_NW\_Nodes(i).X = Rnd() \* W

Goal\_NW\_Nodes(i).Y = Rnd() \* H

Next

drawing()

End Sub

Private Sub EndToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles EndToolStripMenuItem.Click

End

End Sub

Private Sub ConstructNewNetworkToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles ConstructNewNetworkToolStripMenuItem.Click

Timer1.Enabled = False

newnet = 1

i4Goal\_NW\_Nodes = 0

For i = 1 To i4Goal\_NW\_Nodes

For j = 1 To i4Goal\_NW\_Nodes

GoalNetworkAdjancyMatrix(i, j) = False

Next

Next

End Sub

Private Sub AddEdgeToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles AddEdgeToolStripMenuItem.Click

newnet = 2

End Sub

Private Sub AddNodeToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles AddNodeToolStripMenuItem.Click

newnet = 1

End Sub

Private Sub StopToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles StopToolStripMenuItem.Click

newnet = 0

End Sub

Private Sub StopToolStripMenuItem1\_Click(sender As Object, e As EventArgs) Handles StopToolStripMenuItem1.Click

newnet = 0

End Sub

Private Sub PictureBox1\_MouseMove(sender As Object, e As MouseEventArgs) Handles PictureBox1.MouseMove

xm = e.X

ym = e.Y

If MoveNode Then

If clicked And newnet = 0 Then

Goal\_NW\_Nodes(Ti).X = xm

Goal\_NW\_Nodes(Ti).Y = ym

End If

End If

End Sub

Private Sub PictureBox1\_Click(sender As Object, e As EventArgs) Handles PictureBox1.Click

Dim d = 1000

'Dim tx, ty As Integer

For i = 1 To i4Goal\_NW\_Nodes

Dim aa As Integer = xm - Goal\_NW\_Nodes(i).X

Dim bb As Integer = ym - Goal\_NW\_Nodes(i).Y

If Math.Sqrt((aa \* aa) + (bb \* bb)) < d Then

d = Math.Sqrt((aa \* aa) + (bb \* bb))

' tx = Goal\_NW\_Nodes(i).X

' ty = Goal\_NW\_Nodes(i).Y

Ti = i ' Selected Node within the picture

End If

Next

Select Case newnet

Case 0

Case 1 'add node

i4Goal\_NW\_Nodes = i4Goal\_NW\_Nodes + 1

With Goal\_NW\_Nodes(i4Goal\_NW\_Nodes)

.X = xm

.Y = ym

.ID = i4Goal\_NW\_Nodes

End With

Case 2

e1 = Ti

newnet = 3

Case 3

e2 = Ti

AddEdge(Goal\_NW\_Nodes(e1), Goal\_NW\_Nodes(e2))

e1 = 0

e2 = 0

Ti = 0

newnet = 2

End Select

drawing()

'--------------------------------------------------------------------------

If clicked And newnet = 0 Then

clicked = False

Else

clicked = True

End If

End Sub

Private Sub HScrollBar1\_Scroll(sender As Object, e As ScrollEventArgs) Handles HScrollBar1.Scroll

L\_max = HScrollBar1.Value

TextBox3.Text = L\_max

End Sub

Private Sub HScrollBar2\_Scroll(sender As Object, e As ScrollEventArgs) Handles HScrollBar2.Scroll

L\_min = HScrollBar2.Value

TextBox4.Text = L\_min

End Sub

Private Sub ListBox1\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles ListBox1.SelectedIndexChanged

Dim ss() As String = Split(ListBox1.SelectedItem, vbTab)

Dim s As String

For i = 0 To ss.Length - 1

s = s & ":" & ss(i)

Next

MsgBox(s)

End Sub

Private Function GetDD(ByVal type As String) As Integer(,)

Dim DDList(i4Goal\_NW\_Nodes, 2) As Integer ' saving the final list of the degree distribution

Select Case type

Case "in"

Parallel.For(1, i4Goal\_NW\_Nodes, Sub(i)

DDList(i, 1) = Goal\_NW\_Nodes(i).ID

DDList(i, 2) = in\_Degrees(i)

End Sub)

Case "out"

Parallel.For(1, i4Goal\_NW\_Nodes, Sub(i)

DDList(i, 1) = Goal\_NW\_Nodes(i).ID

DDList(i, 2) = out\_Degrees(i)

End Sub)

Case "betweenness"

Dim Bet() As Double = betweeness()

Parallel.For(1, i4Goal\_NW\_Nodes, Sub(i)

DDList(i, 1) = Goal\_NW\_Nodes(i).ID

DDList(i, 2) = Bet(i)

End Sub)

End Select

Return DegreeDistribution(DDList)

End Function

Private Function RefineGoalNetwork() As vertex()

Dim NG(i4Goal\_NW\_Nodes) As vertex

For i = 1 To i4Goal\_NW\_Nodes

NG(i) = Goal\_NW\_Nodes(i)

Next

Return NG

End Function

Private Function DegreeDistribution(ByVal a(,) As Integer) As Integer(,)

'this procedure recieve an array of (node : degree) and return the (degree: distribution) of it.

Dim L As Integer = (a.Length / 3) - 1

Dim DDList(L, 2), Deg, Top As Integer, found As Boolean

For i = 1 To L

Deg = a(i, 2)

found = False

For j = 1 To Top

If DDList(j, 1) = Deg Then

DDList(j, 2) = DDList(j, 2) + 1

found = True

Exit For

End If

Next

If Not found Then

Top = Top + 1

DDList(Top, 1) = Deg

DDList(Top, 2) = 1

End If

Next

Dim DD(Top, 2) As Integer

For i = 0 To Top

DD(i, 1) = DDList(i, 1)

DD(i, 2) = DDList(i, 2)

Next

'Now, sorting...

Dim x1, y1 As Integer

For i = 1 To Top - 1

For j = i + 1 To Top

If DD(j, 1) < DD(i, 1) Then

x1 = DD(i, 1)

y1 = DD(i, 2)

DD(i, 1) = DD(j, 1)

DD(i, 2) = DD(j, 2)

DD(j, 1) = x1

DD(j, 2) = y1

End If

Next

Next

Return DD

End Function

Private Function RandomWithoutZero(ByVal n As Integer) As Integer

Return (Rnd() \* (n - 1)) + 1

End Function

Private Function RandomWithZero(ByVal n As Integer) As Integer

Return Rnd() \* n

End Function

Private Function Sorting(ByVal A() As Integer) As Integer()

Dim l As Integer = A.Length - 1

For i = 0 To l - 1

For j = i + 1 To l

If A(i) > A(j) Then

Dim t As Integer = A(j)

A(j) = A(i)

A(i) = t

End If

Next

Next

Return A

End Function

Private Function in\_Degrees(ByVal s As Integer) As Integer

Dim x As Integer

Parallel.For(1, i4Goal\_NW\_Nodes, Sub(i)

If GoalNetworkAdjancyMatrix(i, s) Then x = x + 1

End Sub)

Return x

End Function

Private Function out\_Degrees(ByVal s As Integer) As Integer

Dim x As Integer

Parallel.For(1, i4Goal\_NW\_Nodes, Sub(i)

If GoalNetworkAdjancyMatrix(s, i) Then x = x + 1

End Sub)

Return x

End Function

Private Function Assortativity() As Double

Dim Top1, Top2, Down1, Down2 As Integer

Dim Pki, Pji As Integer

Dim M As Integer

Parallel.For(1, i4Goal\_NW\_Nodes, Sub(ji)

Pji = undirectedDegree(ji)

For ki = 1 To i4Goal\_NW\_Nodes

Pki = undirectedDegree(ki)

If GoalNetworkAdjancyMatrix(ji, ki) Then

M = M + 1

Top1 = Top1 + (Pji \* Pki)

Top2 = Top2 + (Pki + Pji)

Down1 = Down1 + ((Pji \* Pji) + (Pki \* Pki))

Down2 = Down2 + Pki + Pji

End If

Next

End Sub)

Return ((4 \* M \* Top1) - (Top2 \* Top2)) / ((2 \* M \* Down1) - (Down2 \* Down2))

End Function

Private Function undirectedDegree(ByVal s As Integer) As Integer

Return in\_Degrees(s) + out\_Degrees(s)

End Function

Private Function Distance() As Double

Dim sum As Double = 0

Parallel.For(1, i4Goal\_NW\_Nodes, Sub(i)

For j = 1 To i4Goal\_NW\_Nodes

If j <> i Then

sum = sum + (dijkstra(i, j) / (i4Goal\_NW\_Nodes \* (i4Goal\_NW\_Nodes - 1)))

End If

Next

End Sub)

Return sum

End Function

Private Function Efficiency() As Double

Dim sum As Double = 0

For i = 1 To i4Goal\_NW\_Nodes

Parallel.For(1, i4Goal\_NW\_Nodes, Sub(j)

If j <> i Then

sum = sum + ((1 / (i4Goal\_NW\_Nodes \* (i4Goal\_NW\_Nodes - 1))) \* (1 / dijkstra(i, j)))

End If

End Sub)

Next

Return sum

End Function

Private Function Diameter() As Double

Dim max As Double

For i = 1 To i4Goal\_NW\_Nodes

Parallel.For(1, i4Goal\_NW\_Nodes, Sub(j)

If j <> i Then

Dim d As Double = dijkstra(i, j)

If max < d Then max = d

End If

End Sub)

Next

Return max

End Function

Private Function GetNodeWithID(ByVal ID As Integer) As vertex

Dim x As vertex

Parallel.For(1, i4Goal\_NW\_Nodes, Sub(i)

If Goal\_NW\_Nodes(i).ID = ID Then

x = Goal\_NW\_Nodes(i)

Exit Sub

End If

End Sub)

Return x

End Function

Private Function determine\_attack\_Node(ByVal type As Integer, ByVal Target() As vertex) As vertex

Dim N As Integer = Target.Length - 1

Select Case type

Case 1 ' 1: random

Return Target(RandomWithoutZero(N))

Case 2 'high in degree

Dim AttackedNode As Integer

'herein, we calculate the degrees of all nodes and get the node with highest degree

Dim max As Integer

max = in\_Degrees(Target(1).ID)

AttackedNode = 1

For i = 2 To N

Dim sum As Integer = in\_Degrees(Target(i).ID)

If sum >= max Then

AttackedNode = i

End If

Next

Return Target(AttackedNode)

Case 3 'high out degree

Dim AttackedNode As Integer

'herein, we calculate the degrees of all nodes and get the node with highest degree

Dim max As Integer

max = out\_Degrees(Target(1).ID)

AttackedNode = 1

For i = 2 To N

Dim sum As Integer = out\_Degrees(Target(i).ID)

If sum >= max Then

AttackedNode = i

End If

Next

Return Target(AttackedNode)

Case 4 'high in+out degrees

If Target.Length > 1 Then

Dim AttackedNode As Integer

'herein, we calculate the degrees of all nodes and get the node with highest degree

Dim max As Integer

max = in\_Degrees(Target(1).ID) + out\_Degrees(Target(1).ID)

AttackedNode = 1

For i = 2 To N

Dim sum As Integer = in\_Degrees(Target(i).ID) + out\_Degrees(Target(i).ID)

If sum >= max Then

AttackedNode = i

End If

Next

Return Target(AttackedNode)

End If

Case 5 ' Preferential degree Distribution in

Dim A(,) As Integer = GetDD("in")

Dim T As Integer = Probability\_Distribution(A) ' T: degree of node

Return GetNodeWithID(GetNodeWithThisDegree(T, Target, 1))

Case 6 ' Preferential degree Distribution out

Dim A(,) As Integer = GetDD("out")

Dim T As Integer = Probability\_Distribution(A) ' T: degree of node

Return GetNodeWithID(GetNodeWithThisDegree(T, Target, 2))

Case 7 ' betweeness

Dim max As Double

Dim selected As vertex

Dim A() As Double = betweeness()

For i = 1 To i4Goal\_NW\_Nodes

If A(i) > max Then

max = A(i)

selected = Goal\_NW\_Nodes(i)

End If

Next

Return selected

Case 8 ' PA betweenness

Dim A(,) As Integer = GetDD("betweenness")

Dim T As Integer = Probability\_Distribution(A) ' T: degree of node

Return GetNodeWithID(GetNodeWithThisDegree(T, Target, 3))

End Select

End Function

Private Function GetNodeWithThisDegree(ByVal degree As Integer, ByVal AN() As vertex, ByVal Type As Integer) As Integer

'in this procedure, we will find the approperiate vertex in AN = degree of type (in , out , bet)

'AN () = affected nodes

Dim degrees(AN.Length) As vertex

Dim iDegree As Integer

Select Case Type

Case 1 'in

For i = 1 To AN.Length - 1

If in\_Degrees(AN(i).ID) = degree Then

iDegree = iDegree + 1

degrees(iDegree) = AN(i)

End If

Next

Case 2 'out

For i = 1 To AN.Length - 1

If out\_Degrees(AN(i).ID) = degree Then

iDegree = iDegree + 1

degrees(iDegree) = AN(i)

End If

Next

Case 3 'bet

Dim Bet() As Double = betweeness()

For i = 1 To AN.Length - 1

If Bet(AN(i).ID) = degree Then

iDegree = iDegree + 1

degrees(iDegree) = AN(i)

End If

Next

End Select

Return degrees(RandomWithoutZero(iDegree)).ID

End Function

Private Sub AttackMotif(ByVal a As String)

Select Case a

Case "random"

Dim targettedMotif As Motif

Dim i, j, k As Integer

100:

If i4Goal\_NW\_Nodes >= 3 Then

i = Rnd() \* i4Goal\_NW\_Nodes

j = Rnd() \* i4Goal\_NW\_Nodes

k = Rnd() \* i4Goal\_NW\_Nodes

If i <> j And i <> k And j <> k Then

With targettedMotif

.Node1 = Goal\_NW\_Nodes(i)

.Node2 = Goal\_NW\_Nodes(j)

.node3 = Goal\_NW\_Nodes(k)

End With

delete\_motif(targettedMotif)

End If

Else

'GoTo 100

End If

Case "degree"

Case "betweeness"

End Select

End Sub

Private Sub Surviving(ByVal AN() As vertex, ByVal AE As Integer, ByVal Methodology As Integer)

'-----------------------------------------------------------------------------------------------------------------------

Do While AE > 0

Dim V1, V2 As vertex

'Now selecting the vertices to connect, if they are not connected for sure... :-)

Select Case Methodology

Case 1 ' random

'AN is the affected Nodes, AE is the number of affected edges

Dim r1, r2 As Integer

r1 = RandomWithoutZero(AN.Length - 1)

r2 = RandomWithoutZero(AN.Length - 1)

V1 = AN(r1)

V2 = AN(r2)

Do While r1 = r2 Or GoalNetworkAdjancyMatrix(V1.ID, V2.ID) Or r1 = 0 Or r2 = 0

r1 = RandomWithoutZero(AN.Length - 1)

r2 = RandomWithoutZero(AN.Length - 1)

V1 = AN(r1)

V2 = AN(r2)

Loop

Case 2 ' degree in

'AN is the affected Nodes, AE is the number of affected edges

Dim S(AN.Length - 1) As Integer

For i = 1 To S.Length - 1

S(i) = in\_Degrees(AN(i).ID)

Next

S = Sorting(S)

Dim found As Boolean

For i = 1 To S.Length - 2

For j = i + 1 To S.Length - 1

If Not GoalNetworkAdjancyMatrix(i, j) Or Not GoalNetworkAdjancyMatrix(j, i) Then

V1 = AN(i)

V2 = AN(j)

found = True

Exit For

End If

Next

If found Then Exit For

Next

Case 3 ' degree out

'AN is the affected Nodes, AE is the number of affected edges

Dim S(AN.Length - 1) As Integer

For i = 1 To S.Length - 1

S(i) = out\_Degrees(AN(i).ID)

Next

S = Sorting(S)

Dim found As Boolean

For i = 1 To S.Length - 2

For j = i + 1 To S.Length - 1

If Not GoalNetworkAdjancyMatrix(i, j) Or Not GoalNetworkAdjancyMatrix(j, i) Then

V1 = AN(i)

V2 = AN(j)

found = True

Exit For

End If

Next

If found Then Exit For

Next

Case 4 ' degree (both in and out)

'AN is the affected Nodes, AE is the number of affected edges

Dim S(AN.Length - 1) As Integer

For i = 1 To S.Length - 1

S(i) = in\_Degrees(AN(i).ID) + out\_Degrees(AN(i).ID)

Next

S = Sorting(S)

Dim found As Boolean

For i = 1 To S.Length - 2

For j = i + 1 To S.Length - 1

If Not GoalNetworkAdjancyMatrix(i, j) Or Not GoalNetworkAdjancyMatrix(j, i) Then

V1 = AN(i)

V2 = AN(j)

found = True

Exit For

End If

Next

If found Then Exit For

Next

Case 5 'PA in

'AN is the affected Nodes, AE is the number of affected edges

'Here, we must set the node: degree to an array and then choose one of them.

Dim ND(AN.Length - 1, 2) As Integer ' this is to save Node:in Degree

For i = 1 To AN.Length - 1

ND(i, 1) = AN(i).ID

ND(i, 2) = in\_Degrees(AN(i).ID)

Next

Dim T1, T2 As Integer ' T: degree of node

Dim r1, r2 As Integer ' the id of the selected node

T1 = Probability\_Distribution(DegreeDistribution(ND))

T2 = Probability\_Distribution(DegreeDistribution(ND))

r1 = GetNodeWithThisDegree(T1, AN, 1) 'T1, ND)

r2 = GetNodeWithThisDegree(T2, AN, 1)

V1 = GetNodeWithID(r1)

V2 = GetNodeWithID(r2)

Do While r1 = r2 Or GoalNetworkAdjancyMatrix(V1.ID, V2.ID) Or r1 = 0 Or r2 = 0

T1 = Probability\_Distribution(DegreeDistribution(ND)) ' T: degree of node

T2 = Probability\_Distribution(DegreeDistribution(ND))

r1 = GetNodeWithThisDegree(T1, AN, 1)

r2 = GetNodeWithThisDegree(T2, AN, 1)

V1 = GetNodeWithID(r1)

V2 = GetNodeWithID(r2)

Loop

Case 6 'PA out

'Here, we must set the node: degree to an array and then choose one of them.

'AN is the affected Nodes, AE is the number of affected edges

Dim ND(AN.Length - 1, 2) As Integer

For i = 1 To AN.Length - 1

ND(i, 1) = AN(i).ID

ND(i, 2) = out\_Degrees(AN(i).ID)

Next

Dim T1, T2 As Integer

Dim r1, r2 As Integer

T1 = Probability\_Distribution(DegreeDistribution(ND)) ' T: degree of node

T2 = Probability\_Distribution(DegreeDistribution(ND))

r1 = GetNodeWithThisDegree(T1, AN, 2)

r2 = GetNodeWithThisDegree(T2, AN, 2)

V1 = GetNodeWithID(r1)

V2 = GetNodeWithID(r2)

Do While r1 = r2 Or GoalNetworkAdjancyMatrix(V1.ID, V2.ID) Or r1 = 0 Or r2 = 0

T1 = Probability\_Distribution(DegreeDistribution(ND)) ' T: degree of node

T2 = Probability\_Distribution(DegreeDistribution(ND))

r1 = GetNodeWithThisDegree(T1, AN, 2)

r2 = GetNodeWithThisDegree(T2, AN, 2)

V1 = GetNodeWithID(r1)

V2 = GetNodeWithID(r2)

Loop

Case 7

'betweenness

'AN is the affected Nodes, AE is the number of affected edges

'Here, we must set the node: degree to an array and then choose one of them.

Dim ND(AN.Length - 1, 2) As Integer

Dim Bet() As Double = betweeness()

For i = 1 To AN.Length - 1

ND(i, 1) = AN(i).ID

ND(i, 2) = Bet(AN(i).ID)

Next

Dim T1, T2 As Integer

Dim r1, r2 As Integer

T1 = Probability\_Distribution(DegreeDistribution(ND)) ' T: degree of node

T2 = Probability\_Distribution(DegreeDistribution(ND))

r1 = GetNodeWithThisDegree(T1, AN, 3)

r2 = GetNodeWithThisDegree(T2, AN, 3)

V1 = GetNodeWithID(r1)

V2 = GetNodeWithID(r2)

Do While r1 = r2 Or GoalNetworkAdjancyMatrix(V1.ID, V2.ID) Or r1 = 0 Or r2 = 0

T1 = Probability\_Distribution(DegreeDistribution(ND)) ' T: degree of node

T2 = Probability\_Distribution(DegreeDistribution(ND))

r1 = GetNodeWithThisDegree(T1, AN, 3)

r2 = GetNodeWithThisDegree(T2, AN, 3)

V1 = GetNodeWithID(r1)

V2 = GetNodeWithID(r2)

Loop

Case 8

'betweenness

'AN is the affected Nodes, AE is the number of affected edges

'Here, we must set the node: degree to an array and then choose one of them.

Dim ND(AN.Length - 1, 2) As Integer

Dim Bet() As Double = betweeness()

For i = 1 To AN.Length - 1

ND(i, 1) = AN(i).ID

ND(i, 2) = Bet(AN(i).ID)

Next

Dim T1, T2 As Integer

Dim r1, r2 As Integer

T1 = Probability\_Distribution(DegreeDistribution(ND)) ' T: degree of node

T2 = Probability\_Distribution(DegreeDistribution(ND))

r1 = GetNodeWithThisDegree(T1, AN, 3)

r2 = GetNodeWithThisDegree(T2, AN, 3)

V1 = GetNodeWithID(r1)

V2 = GetNodeWithID(r2)

Do While r1 = r2 Or GoalNetworkAdjancyMatrix(V1.ID, V2.ID) Or r1 = 0 Or r2 = 0

T1 = Probability\_Distribution(DegreeDistribution(ND)) ' T: degree of node

T2 = Probability\_Distribution(DegreeDistribution(ND))

r1 = GetNodeWithThisDegree(T1, AN, 3)

r2 = GetNodeWithThisDegree(T2, AN, 3)

V1 = GetNodeWithID(r1)

V2 = GetNodeWithID(r2)

Loop

End Select

AddEdge(V1, V2)

AE = AE - 1

Loop

End Sub

Private Function Same(ByVal v1 As vertex, ByVal v2 As vertex) As Boolean

If v1.ID = v2.ID And v1.Numeric\_ID = v2.Numeric\_ID And v1.String\_ID = v2.String\_ID Then

Return True

Else

Return False

End If

End Function

Public Sub RemoveUnconnectedNodes()

Dim xin(i4Goal\_NW\_Nodes), xout(i4Goal\_NW\_Nodes) As Integer

For i = 1 To i4Goal\_NW\_Nodes

xin(i) = in\_Degrees(i)

xout(i) = out\_Degrees(i)

If xin(i) = 0 And xout(i) = 0 Then

delete\_node(Goal\_NW\_Nodes(i))

End If

Next

End Sub

Private Sub delete\_motif(ByVal M As Motif)

'now we must delete every node seperatly and remove the edges connecting to that nodes

Dim A, B, C As vertex

A = M.Node1

B = M.Node2

C = M.node3

delete\_node(A)

delete\_node(B)

delete\_node(C)

End Sub

Private Sub delete\_node(ByVal V As vertex)

'before we deleting the node, we must delete all connection edges...

' we simply exchange the list of connection to that node with the last list of connection and simply reduce it by 1...

'then, we can delete the node using the following code.

'first steps:

'get the sequence of the vertex within the vertex set:

Dim seq As Integer

For k As Integer = 1 To i4Goal\_NW\_Nodes

If Equals(Goal\_NW\_Nodes(k), V) Then

seq = k

Dim temp As Boolean

For i As Integer = 1 To i4Goal\_NW\_Nodes

temp = GoalNetworkAdjancyMatrix(k, i)

GoalNetworkAdjancyMatrix(k, i) = GoalNetworkAdjancyMatrix(i4Goal\_NW\_Nodes, i)

GoalNetworkAdjancyMatrix(i4Goal\_NW\_Nodes, i) = temp

Next

For i As Integer = 1 To i4Goal\_NW\_Nodes

temp = GoalNetworkAdjancyMatrix(i, k)

GoalNetworkAdjancyMatrix(i, k) = GoalNetworkAdjancyMatrix(i, i4Goal\_NW\_Nodes)

GoalNetworkAdjancyMatrix(i, i4Goal\_NW\_Nodes) = temp

Next

End If

Next

If Equals(Goal\_NW\_Nodes(seq), V) Then

Dim tempV As vertex

tempV = Goal\_NW\_Nodes(i4Goal\_NW\_Nodes)

Goal\_NW\_Nodes(i4Goal\_NW\_Nodes) = V

Goal\_NW\_Nodes(seq) = tempV

'Goal\_NW\_Nodes(seq).ID = V.ID

i4Goal\_NW\_Nodes = i4Goal\_NW\_Nodes - 1

End If

End Sub

Private Sub showdegrees()

Dim tt As String

For i = 1 To i4Goal\_NW\_Nodes

tt = tt & vbTab & Goal\_NW\_Nodes(i).ID & vbTab & in\_Degrees(Index(Goal\_NW\_Nodes(i))) & " : " & out\_Degrees(Index(Goal\_NW\_Nodes(i))) & vbNewLine

Next

TextBox2.Text = tt

End Sub

Private Function dijkstra(ByVal s As Integer, ByVal t As Integer) As Integer

Const infinity = 8000000

Dim current As Integer

Dim distance(i4Goal\_NW\_Nodes) As Double

Dim perm(i4Goal\_NW\_Nodes) As String

Dim precede(i4Goal\_NW\_Nodes) As Double

For vv = 1 To i4Goal\_NW\_Nodes

distance(vv) = infinity

perm(vv) = "nonmember"

Next

perm(s) = "member"

distance(s) = 0

current = s

While current <> t

Dim dc As Integer = distance(current)

For i = 1 To i4Goal\_NW\_Nodes

If GoalNetworkAdjancyMatrix(current, i) Then

Dim newdist As Integer = dc + getWeight(GoalNetworkAdjancyMatrix(current, i))

If newdist < distance(i) Then

distance(i) = newdist

precede(i) = current

End If

End If

Next

k = 0

Dim smallest As Double = infinity

For i = 1 To i4Goal\_NW\_Nodes

If perm(i) = "nonmember" Then

If distance(i) < smallest Then

k = i

smallest = distance(i)

End If

End If

Next

If k = 0 Then

Exit While

End If

current = k

perm(k) = "member"

End While

For i = 1 To i4Goal\_NW\_Nodes

If distance(i) = infinity Then distance(i) = 0

Next

Return distance(t)

End Function

Private Function getWeight(ByVal x As Boolean) As Integer

If x Then

Return 1

Else

Return 0

End If

End Function

Private Function Average\_geodesic\_path() As Double

Dim sum As Integer

For i = 1 To i4Goal\_NW\_Nodes

For j = 1 To i4Goal\_NW\_Nodes

If i <> j Then

sum = sum + dijkstra(i, j)

End If

Next

Next

Return sum / (i4Goal\_NW\_Nodes \* (i4Goal\_NW\_Nodes - 1))

End Function

Private Sub InverseAverageGeodesicPathToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles InverseAverageGeodesicPathToolStripMenuItem.Click

MsgBox(Average\_geodesic\_path())

End Sub

Private Sub ShowShorestPathesToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles ShowShorestPathesToolStripMenuItem.Click

Dim txt As String

For i = 1 To i4Goal\_NW\_Nodes

txt = txt & i & " : "

For j = 1 To i4Goal\_NW\_Nodes

txt = txt & CStr(dijkstra(i, j)) & " : "

Next

txt = txt & vbNewLine

Next

MsgBox(txt)

End Sub

Private Function Clustering\_Coeffecient() As Double

Dim c As Double

For i = 1 To i4Goal\_NW\_Nodes

c = c + CC(i)

Next

Return c / i4Goal\_NW\_Nodes

End Function

Private Function CC(ByVal s As Integer) As Double

Dim Neighbor(i4Goal\_NW\_Nodes) As Integer, TopN As Integer

For i = 1 To i4Goal\_NW\_Nodes

If GoalNetworkAdjancyMatrix(i, s) Or GoalNetworkAdjancyMatrix(s, i) Then

TopN = TopN + 1

Neighbor(TopN) = i

End If

Next

Dim sum As Integer = 0

For i = 1 To TopN - 1

Dim x, y As Integer

x = Neighbor(i)

For j = i + 1 To TopN

y = Neighbor(j)

If GoalNetworkAdjancyMatrix(x, y) Or GoalNetworkAdjancyMatrix(y, x) Then

sum = sum + 1

End If

Next

Next

Return sum \* 2 / (i4Goal\_NW\_Nodes \* (i4Goal\_NW\_Nodes - 1))

End Function

Private Sub ClusteringCoefficientToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles ClusteringCoefficientToolStripMenuItem.Click

MsgBox(Clustering\_Coeffecient())

End Sub

Private Sub AssortativityToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles AssortativityToolStripMenuItem.Click

MsgBox(Assortativity)

End Sub

Private Function NumberOfEdges() As Integer

Dim ss As Integer

For i = 1 To i4Goal\_NW\_Nodes

ss = ss + in\_Degrees(i)

Next

NumberOfEdges = ss \* 2

End Function

Private Sub Strategy(ByVal TypeofAttack As Integer, ByVal TypeofAdd As Integer, ByVal TypeofMethodology As Integer)

Dim TargetNode As vertex

'TypeofAttack -----> 1: random, 2: high degrees, 3: Preferential degree Distribution, 4: Betweenness PA distribution

'TypeofAdd --------> 1: all , 2: affected only

'TypeofMethodology-> 1: random, 2: degree 3: PA, 4: Bet, 5: Bet PA, 6: Closeness, 7: Close PA, 8: Clustering Coeffecient

'----------------------------------------------------------------------------------------------------------------

'Form2.addToGrid(Get\_ComNet\_Features())

Text = Text & vbNewLine & Get\_ComNet\_Features()

TargetNode = determine\_attack\_Node(TypeofAttack, GetNodesWithDegree(Goal\_NW\_Nodes))

Dim AN() As vertex = Affected\_Nodes(TargetNode, TypeofAdd)

Do While AN Is Nothing Or TargetNode.ID = 0

TargetNode = determine\_attack\_Node(TypeofAttack, GetNodesWithDegree(Goal\_NW\_Nodes))

If TargetNode.ID <> 0 Then

AN = Affected\_Nodes(TargetNode, TypeofAdd)

Else

If TypeofAttack = 8 Or TypeofAttack = 7 Then

TargetNode = Goal\_NW\_Nodes(RandomWithoutZero(i4Goal\_NW\_Nodes))

AN = Affected\_Nodes(TargetNode, TypeofAdd)

End If

End If

Loop

Dim AE As Integer = DeleteEdges(TargetNode)

Surviving(AN, AE, TypeofMethodology)

End Sub

Private Sub Attacking(ByVal TypeofAttack As Integer, ByVal TypeofAdd As Integer, ByVal TypeofMethodology As Integer)

Dim TargetNode As vertex

'TypeofAttack -----> 1: random, 2: high degrees, 3: Preferential degree Distribution, 4: Betweenness PA distribution

'TypeofAdd --------> 1: all , 2: affected only

'TypeofMethodology-> 1: random, 2: degree 3: PA, 4: Bet, 5: Bet PA, 6: Closeness, 7: Close PA, 8: Clustering Coeffecient

'----------------------------------------------------------------------------------------------------------------

'Form2.addToGrid(Get\_ComNet\_Features())

Text = Text & vbNewLine & Get\_ComNet\_Features()

TargetNode = determine\_attack\_Node(TypeofAttack, GetNodesWithDegree(Goal\_NW\_Nodes))

Dim AN() As vertex = Affected\_Nodes(TargetNode, TypeofAdd)

Do While AN Is Nothing Or TargetNode.ID = 0

TargetNode = determine\_attack\_Node(TypeofAttack, GetNodesWithDegree(Goal\_NW\_Nodes))

If TargetNode.ID <> 0 Then

AN = Affected\_Nodes(TargetNode, TypeofAdd)

Else

If TypeofAttack = 8 Or TypeofAttack = 7 Then

TargetNode = Goal\_NW\_Nodes(RandomWithoutZero(i4Goal\_NW\_Nodes))

AN = Affected\_Nodes(TargetNode, TypeofAdd)

End If

End If

Loop

Dim AE As Integer = DeleteEdges(TargetNode)

End Sub

Private Function GetNodesWithDegree(ByVal A() As vertex) As vertex()

Dim N As Integer = A.Length

Dim B(N) As vertex

Dim iB As Integer

For i = 1 To N - 1

If in\_Degrees(A(i).ID) > 0 Or out\_Degrees(A(i).ID) > 0 Then

iB += 1

B(iB) = A(i)

End If

Next

Dim T(iB) As vertex

For i = 1 To iB

T(i) = B(i)

Next

Return T

End Function

Private Function DeleteEdges(ByVal V As vertex) As Integer

Dim x As Integer

Parallel.For(1, i4Goal\_NW\_Nodes, Sub(i)

If GoalNetworkAdjancyMatrix(i, V.ID) Then

GoalNetworkAdjancyMatrix(i, V.ID) = False

DeleteEdges += 1

End If

If GoalNetworkAdjancyMatrix(V.ID, i) Then

GoalNetworkAdjancyMatrix(V.ID, i) = False

DeleteEdges += 1

End If

End Sub)

End Function

Private Function Affected\_Nodes(ByVal V As vertex, ByVal TypeofAddition As Integer) As vertex()

'1: all , 2: affected only

Select Case TypeofAddition

Case 1

Return Goal\_NW\_Nodes

Case 2 'affected only

'all nodes that have connection with deleted node V.

Dim AN(i4Goal\_NW\_Nodes) As vertex

Dim iAN As Integer 'number of affected nodes

For i = 1 To i4Goal\_NW\_Nodes

If (GoalNetworkAdjancyMatrix(i, V.ID) Or GoalNetworkAdjancyMatrix(V.ID, i)) And i <> V.ID Then

iAN = iAN + 1

AN(iAN) = Goal\_NW\_Nodes(i)

End If

Next

If iAN <> 0 Then

Dim SAN(iAN + 1) As vertex

For i = 1 To iAN

SAN(i) = AN(i)

Next

SAN(iAN + 1) = V ' here we added the deleted node to the affected nodes to be considered for reconstruction :-D

Return SAN

End If

End Select

End Function

Private Function Network\_Stand() As Boolean

Dim t As Integer = NumberOfEdges()

Dim stand As Boolean

'---------------------------

If i4Goal\_NW\_Nodes > 0 And t > 0 Then

stand = True

Else

stand = False

End If

Return stand

End Function

Private Function Get\_ComNet\_Features() As String

Dim prop As String

'prop = i4Goal\_NW\_Nodes

'prop = prop & vbTab & NumberOfEdges()

prop = Assortativity()

prop = prop & vbTab & Clustering\_Coeffecient()

prop = prop & vbTab & Average\_geodesic\_path()

prop = prop & vbTab & distance()

prop = prop & vbTab & Efficiency()

prop = prop & vbTab & Diameter()

Return prop

'Averaged\_Diameter() 'of the network

End Function

Private Function betweeness() As Double()

'find the betweeness of node with index i

Dim Cb(i4Goal\_NW\_Nodes) As Double

Dim S As New Stack

Dim P(i4Goal\_NW\_Nodes) As MyList

Dim sigma(i4Goal\_NW\_Nodes) As Double

Dim d(i4Goal\_NW\_Nodes) As Double

Dim Q As New Queue(Of vertex)

'-------------------------------------

ResetIndeces()

'------------------------------------

For iV As Integer = 1 To i4Goal\_NW\_Nodes

Cb(iV) = 0

Next

For SS = 1 To i4Goal\_NW\_Nodes

S.Clear()

For iW = 1 To i4Goal\_NW\_Nodes

ReDim P(iW).PList(i4Goal\_NW\_Nodes)

P(iW).iP = 0

sigma(iW) = 0 't=w

d(iW) = -1 't=w

Next

sigma(SS) = 1

d(SS) = 0

Q.Clear()

Q.Enqueue(Goal\_NW\_Nodes(SS))

While Q.Count > 0

Dim v As vertex = Q.Dequeue

S.Push(v)

Dim iV As Integer = v.ID

For iW = 1 To i4Goal\_NW\_Nodes

If GoalNetworkAdjancyMatrix(iV, iW) Then

If d(iW) < 0 Then

Q.Enqueue(Goal\_NW\_Nodes(iW))

d(iW) = d(iV) + 1

End If

If d(iW) = d(iV) + 1 Then

sigma(iW) = sigma(iW) + sigma(iV)

P(iW).iP = P(iW).iP + 1

P(iW).PList(P(iW).iP) = v

End If

End If

Next

End While

Dim Gamma(i4Goal\_NW\_Nodes) As Integer

While S.Count > 0

Dim MyW As vertex = S.Pop()

Dim iw As Integer = MyW.ID

For i = 0 To P(iw).iP

Dim v As vertex = P(iw).PList(i)

Dim iV As Integer

iV = v.ID

Gamma(iV) = Gamma(iV) + ((1 + Gamma(iw)) \* sigma(iV) / sigma(iw))

If iw <> SS Then

Cb(iw) = Cb(iw) + Gamma(iw)

End If

Next

End While

Next

Return Cb

End Function

Private Sub ResetIndeces()

Parallel.For(1, i4Goal\_NW\_Nodes, Sub(i)

Goal\_NW\_Nodes(i).ID = Index(Goal\_NW\_Nodes(i))

End Sub)

End Sub

Private Sub BetweennessCentralityToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles BetweennessCentralityToolStripMenuItem.Click

Dim t As String

Dim cb() As Double = betweeness()

For i = 1 To i4Goal\_NW\_Nodes

Form2.DataGridView1.Rows.Add(New String() {Goal\_NW\_Nodes(i).String\_ID, cb(i) / 2})

Next

Form2.Show()

End Sub

Private Sub ShowEdgesToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles ShowEdgesToolStripMenuItem.Click

showEdges()

End Sub

Private Sub showEdges()

Dim t As String

For i = 1 To i4Goal\_NW\_Nodes

For j = 1 To i4Goal\_NW\_Nodes

If GoalNetworkAdjancyMatrix(i, j) Then

Form2.DataGridView1.Rows.Add(New String() {Goal\_NW\_Nodes(i).String\_ID, Goal\_NW\_Nodes(j).String\_ID})

End If

Next

Next

Form2.Show()

End Sub

Public Function Probability\_Distribution(ByVal A(,) As Integer) As Integer

'Probability distribution recieve an array of degree: distribution

'and select random value among them and return the

'selected degree, So that the array A is the degrees set

'and later the system has to pick up the selected degree randomly.

Dim SD As Integer

Dim Total As Integer

Dim E As Integer = (A.Length / 3) - 1

For i = 1 To E

Total = Total + A(i, 2)

Next

If Total = 0 Then

MsgBox("please specify the network first! ")

Return 0

End If

Dim oldT As Double = 0

Dim R As Double

R = Rnd(1)

For i As Integer = 1 To E

Dim R1, R2 As Double

R1 = oldT

R2 = oldT + (A(i, 2) / Total)

If R >= R1 And R < R2 Then

Return A(i, 1)

Else

oldT = oldT + (A(i, 2) / Total)

End If

Next

Return SD

End Function

Private Sub IndegreeToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles IndegreeToolStripMenuItem.Click

Dim a(,) As Integer = GetDD("in")

For i = 1 To (a.Length / 3) - 1

Form2.DataGridView1.Rows.Add(New String() {a(i, 1), a(i, 2)})

Next

Form2.Show()

End Sub

Private Sub OutdegreeToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles OutdegreeToolStripMenuItem.Click

Dim a(,) As Integer = GetDD("out")

For i = 1 To (a.Length / 3) - 1

Form2.DataGridView1.Rows.Add(New String() {a(i, 1), a(i, 2)})

Next

Form2.Show()

End Sub

Private Sub BetweennessToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles BetweennessToolStripMenuItem.Click

Dim a(,) As Integer = GetDD("betweenness")

For i = 1 To (a.Length / 3) - 1

Form2.DataGridView1.Rows.Add(New String() {a(i, 1), a(i, 2)})

Next

Form2.Show()

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

If Timer2.Enabled Then

Timer2.Enabled = False

Else

Timer2.Enabled = True

End If

End Sub

Private Function TotalEdges() As Integer

Dim M As Integer = 0

Parallel.For(1, i4Goal\_NW\_Nodes, Sub(i)

For j = 1 To i4Goal\_NW\_Nodes

If GoalNetworkAdjancyMatrix(i, j) Then M = M + 1

Next

End Sub)

Return M

End Function

Private Sub RadioButton1\_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButton1.CheckedChanged

If RadioButton1.Enabled Then

MoveNode = True

Else

MoveNode = False

End If

End Sub

Private Sub RadioButton2\_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButton2.CheckedChanged

If RadioButton1.Enabled Then

MoveNode = True

Else

MoveNode = False

End If

End Sub

Private Sub Timer2\_Tick(sender As Object, e As EventArgs) Handles Timer2.Tick

Dim add, att, met As Integer

If RadioButton1.Checked Then att = 1

If RadioButton2.Checked Then att = 2

If RadioButton3.Checked Then att = 3

If RadioButton4.Checked Then att = 4

If RadioButton5.Checked Then att = 5

If RadioButton6.Checked Then att = 6

If RadioButton7.Checked Then add = 1

If RadioButton8.Checked Then add = 2

If RadioButton9.Checked Then met = 1

If RadioButton10.Checked Then met = 2

If RadioButton11.Checked Then met = 3

If RadioButton12.Checked Then met = 4

If RadioButton13.Checked Then met = 5

If RadioButton14.Checked Then met = 6

Dim TargetNode As vertex = determine\_attack\_Node(att, Goal\_NW\_Nodes) ' GetNodeWithID(Ti)

'3) Get Affected Nodes AN and Edges, each edges connects bet deleted and affected reconnected again.

'5) delete the node and connected edges

'in\_Degrees(TargetNode.ID) + out\_Degrees(TargetNode.ID)

Dim AN() As vertex = Affected\_Nodes(TargetNode, add)

Dim AE As Integer = DeleteEdges(TargetNode)

'4) apply surviving methodologies

Surviving(AN, AE, met)

drawing()

Label2.Text = "Edges: " & TotalEdges()

End Sub

Private Sub ApplyingStrategyToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles ApplyingStrategyToolStripMenuItem.Click

SFD.Title = "Simulation Location Specification"

SFD.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*"

SFD.ShowDialog()

Dim path As String = SFD.FileName

ProgressBar1.Maximum = 6 \* 2

Simulating = True

NumberofSimulation = i4Goal\_NW\_Nodes / 2

For ToAtt = 1 To 8 'TypeOfAttack

For ToAdd = 1 To 2 'TyoeofAdd

For Meth = 1 To 8 'Surviving Methodology

Dim NPath As String = path

NPath &= "\_" & ToAtt & "\_" & ToAdd & "\_" & Meth & ".txt"

LoadNetwork(ToolStripStatusLabel1.Text)

Text = "Assortativity" & vbTab & "Correlation" & vbTab & "Clustering\_Coeffecient" & vbTab & "connected\_cluster" \_

& vbTab & "Average\_geodesic\_path" & vbTab & "distance" & vbTab & "Efficiency" & vbTab & "Diameter"

'------------------------------------------------------------------------------------------------------------------

For i = 1 To NumberofSimulation ' 100 is number of simulation of attacking and surviving

Strategy(ToAtt, ToAdd, Meth)

Next

'Saving result to specific path

My.Computer.FileSystem.WriteAllText(NPath, Text, False)

Next

'ProgressBar1.Value += 1

Next

Next

Simulating = False

MsgBox("Simulation is done.")

'Form2.Show()

End Sub

Private Sub PardingResultsToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles PardingResultsToolStripMenuItem.Click

Dim path As String

With OFD1

.Title = "Simulation Location Specification"

.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*"

.ShowDialog()

path = .FileName

End With

Dim A() As String = path.Split("\")

Dim B() As String = A(A.Length - 1).Split(".")

Dim n() As String = B(1).Split("\_")

Dim ss As String

For i = 1 To n.Length - 1

ss = ss & vbTab & n(i)

Next

Dim newPath As String

For i = 0 To A.Length - 2

newPath = newPath & A(i) & "\"

Next

Dim S1, S2, S3, S4, S5, S6, S7, S8 As String

S1 = newPath & "\Assortativity"

S2 = newPath & "\Clustering Coefficient"

S3 = newPath & "\Averaged Geodesic Path"

S4 = newPath & "\Distance"

S5 = newPath & "\Efficiency"

S6 = newPath & "\Diameter"

NewFile(S1)

NewFile(S2)

NewFile(S3)

NewFile(S4)

NewFile(S5)

NewFile(S6)

newPath = newPath & B(0) & ".txt\_"

Dim np As String

For i = 1 To 8

For j = 1 To 2

Dim T1(,) As String = ConvertTextAsMatrix(ReadFile(newPath & i & "\_" & j & "\_" & 1 & ".txt"))

Dim T2(,) As String = ConvertTextAsMatrix(ReadFile(newPath & i & "\_" & j & "\_" & 2 & ".txt"))

Dim T3(,) As String = ConvertTextAsMatrix(ReadFile(newPath & i & "\_" & j & "\_" & 3 & ".txt"))

Dim T4(,) As String = ConvertTextAsMatrix(ReadFile(newPath & i & "\_" & j & "\_" & 4 & ".txt"))

Dim T5(,) As String = ConvertTextAsMatrix(ReadFile(newPath & i & "\_" & j & "\_" & 5 & ".txt"))

Dim T6(,) As String = ConvertTextAsMatrix(ReadFile(newPath & i & "\_" & j & "\_" & 6 & ".txt"))

Dim T7(,) As String = ConvertTextAsMatrix(ReadFile(newPath & i & "\_" & j & "\_" & 7 & ".txt"))

Dim T8(,) As String = ConvertTextAsMatrix(ReadFile(newPath & i & "\_" & j & "\_" & 8 & ".txt"))

Dim F1(,) As String = GetNewFormat(0, T1, T2, T3, T4, T5, T6, T7, T8)

Dim F2(,) As String = GetNewFormat(1, T1, T2, T3, T4, T5, T6, T7, T8)

Dim F3(,) As String = GetNewFormat(2, T1, T2, T3, T4, T5, T6, T7, T8)

Dim F4(,) As String = GetNewFormat(3, T1, T2, T3, T4, T5, T6, T7, T8)

Dim F5(,) As String = GetNewFormat(4, T1, T2, T3, T4, T5, T6, T7, T8)

Dim F6(,) As String = GetNewFormat(5, T1, T2, T3, T4, T5, T6, T7, T8)

'Dim F7(,) As String = GetNewFormat(6, T1, T2, T3, T4, T5, T6, T7, T8)

'Dim F8(,) As String = GetNewFormat(7, T1, T2, T3, T4, T5, T6, T7, T8)

saving(S1 & "\" & i & "\_" & j & "\_" & "\_Assortativity.txt", ConvertMatrixToText(F1))

saving(S2 & "\" & i & "\_" & j & "\_" & "\_ClusteringCoefficient.txt", ConvertMatrixToText(F2))

saving(S3 & "\" & i & "\_" & j & "\_" & "\_AveragedGeodesicPath.txt", ConvertMatrixToText(F3))

saving(S4 & "\" & i & "\_" & j & "\_" & "\_distance.txt", ConvertMatrixToText(F4))

saving(S5 & "\" & i & "\_" & j & "\_" & "\_Efficiency.txt", ConvertMatrixToText(F5))

saving(S6 & "\" & i & "\_" & j & "\_" & "\_Diameter.txt", ConvertMatrixToText(F6))

'saving(newPath & i & "\_" & j & "\_" & "\_N.txt", ConvertMatrixToText(F7))

'saving(newPath & i & "\_" & j & "\_" & "\_N.txt", ConvertMatrixToText(F8))

Next

Next

MsgBox("Parsing is done!..")

End Sub

Private Sub NewFile(ByVal YourPath)

If (Not System.IO.Directory.Exists(YourPath)) Then

System.IO.Directory.CreateDirectory(YourPath)

End If

End Sub

Private Function GetNewFormat(ByVal Col As Integer, ByVal t1(,) As String, ByVal t2(,) As String, ByVal t3(,) As String, ByVal t4(,) As String,

ByVal t5(,) As String, ByVal t6(,) As String, ByVal t7(,) As String, ByVal t8(,) As String) As String(,)

NumberofSimulation = i4Goal\_NW\_Nodes / 2

Dim X(NumberofSimulation, 8) As String

X(0, 0) = "Random"

X(0, 1) = "AHIn"

X(0, 2) = "AHOut"

X(0, 3) = "AHInOut"

X(0, 4) = "PAIn"

X(0, 5) = "PAOut"

X(0, 6) = "AHB"

X(0, 7) = "PABet"

For i = 1 To NumberofSimulation - 1

X(i, 0) = t1(i, Col)

X(i, 1) = t2(i, Col)

X(i, 2) = t3(i, Col)

X(i, 3) = t4(i, Col)

X(i, 4) = t5(i, Col)

X(i, 5) = t6(i, Col)

X(i, 6) = t7(i, Col)

X(i, 7) = t8(i, Col)

Next

Return X

End Function

Private Function ConvertMatrixToText(ByVal A(,) As String) As String

Dim Text As String = ""

For i = 0 To NumberofSimulation - 1

For j = 0 To 8

If j = 0 Then

Text = Text & A(i, j)

Else

Text = Text & vbTab & A(i, j)

End If

Next

Text = Text & vbNewLine

Next

Return Text

End Function

Private Function ConvertTextAsMatrix(ByVal Text As String) As String(,)

Dim lines() As String = Text.Split(vbNewLine)

Dim Rows As Integer = lines.Length

Dim Columns As Integer = lines(1).Split(vbTab).Length

Dim R(Rows, Columns) As String

For i = 0 To Rows - 1

Dim B() As String = lines(i).Split(vbTab)

For j = 0 To Columns - 1

R(i, j) = B(j)

Next

Next

Return R

End Function

Private Sub Button4\_Click(sender As Object, e As EventArgs) Handles Button4.Click

End

End Sub

Private Sub AttackingPerformenceToolStripMenuItem\_Click(sender As Object, e As EventArgs) Handles AttackingPerformenceToolStripMenuItem.Click

SFD.Title = "Simulation Location Specification"

SFD.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*"

SFD.ShowDialog()

Dim path As String = SFD.FileName

ProgressBar1.Maximum = 6 \* 2

Simulating = True

NumberofSimulation = i4Goal\_NW\_Nodes / 2

Dim ToAdd As Integer = 1

For ToAtt = 1 To 8 'TypeOfAttack

Dim Meth As Integer = 1

Dim NPath As String = path

NPath &= "\_" & ToAtt & "\_" & ToAdd & "\_" & Meth & ".txt"

LoadNetwork(ToolStripStatusLabel1.Text)

Text = "Assortativity" & vbTab & "Correlation" & vbTab & "Clustering\_Coeffecient" & vbTab & "connected\_cluster" \_

& vbTab & "Average\_geodesic\_path" & vbTab & "distance" & vbTab & "Efficiency" & vbTab & "Diameter"

'------------------------------------------------------------------------------------------------------------------

For i = 1 To NumberofSimulation ' 100 is number of simulation of attacking and surviving

Attacking(ToAtt, ToAdd, Meth)

Next

'Saving result to specific path

My.Computer.FileSystem.WriteAllText(NPath, Text, False)

'ProgressBar1.Value += 1

Next

Simulating = False

MsgBox("Simulation is done.")

'Form2.Show()

End Sub

End Class