Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

Software Development on Soft Graph Matrix

M.G.Karunambigai[1] and V. Gnaneswaran[2]
[1] Associate Professor & Head, PG & Research Department of Mathematics,
Sri Vasavi College, Erode - 638 316, Tamilnadu, India.
[2] Assistant Professor, Department of Mathematics, Nandha Arts and Science College (Autonomous), Erode - 638 052,
Tamilnadu, India.

***Abstract* – In this paper MATLAB-based software tool has been developed for the soft graph matrices. This system allows the key operations such as the Cartesian product, AND, OR, and complement on soft graph matrices. The effectiveness and applicability of the above operations in soft graph matrices are validated through illustrations and a real – world application.**

Keywords – Soft Set, Soft Graph, Soft Graph Matrix, MATLAB Algorithm.

1. INTRODUCTION

Uncertainty is the unavoidable challenge in several fields, and the mathematical models are proposed to manage it. By addressing this uncertainty, [1] introduced the soft set theory, it is the more general and the adaptable framework which avoids the reliance on additional structures such as the membership functions or probabilities.

On this thought, the study [2] extended concept into the graph theory with the introduction of the soft graphs. Since the pioneering work of Euler [3], the graph theory plays the major role to solve the problems in computer science, operations research, and network modeling[4-7]. The incorporation of the soft sets with graph theory provided the better way to represent and analyze uncertain relationships in the complex systems.

In the computational practice, graph structures are represented by utilizing the matrices like, the adjacency matrices, that translate the graphs into numerical form for better analysis. Although, the standard matrix operations are not equipped to handle the uncertainty. To fill this, soft graph matrices were proposed. The key advancement was made by Karunambigai and Gnaneswaran[8], who defined the operations such as the AND, OR, Cartesian product, and complement for the soft graphs, and shown their practical relevance in the decision-making issues.

Also theory of soft graph matrices has the considerable development, but the practical computational tools for implementation of these ideas were still lacking. To bridge this gap, this paper present the MATLAB-based software system which enables construction, manipulation and analysis of soft graph matrices. The system support the important operations like, the AND, OR, Cartesian product, and complement, and extending its capabilities with the additional analytical features. This paper provides the researchers and practitioners with the valuable tool for advancing studies and the real-world applications in uncertain data environments.

2. PRELIMINARIES

Definition 2.1

Let G = (V, E) be the simple graph, A any nonempty set. Let R be arbitrary relation between elements of $A$and$V$. (i.e.)R$\subseteq A \times V$. A set valued function F: A$\rightarrow$ $P(V)$is defined as F(x)= $\{y \in V / x R y\}$.

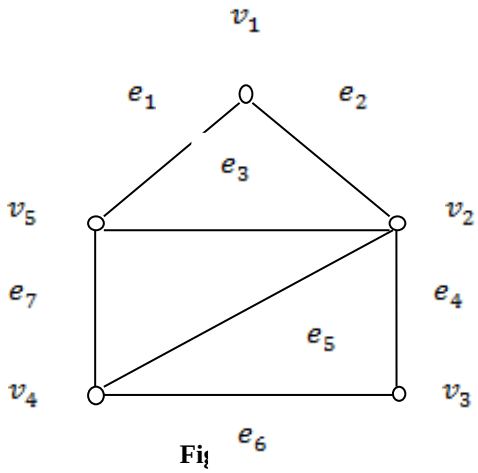
The pair (F,A) is soft set over V.Then (F,A) is said to be a Soft graph of G if the subgraph induced by F(x) in G($F^{\%}$(x)) is a connected subgraph of G for all x ∈ A.

The set of all soft graph of G is denoted by SG(G).

Example 2.2

Consider simple graph G = (V, E) as shown in Figure.1.

Let, V = {v_1, v_2, v_3, v_4, v_5 } be a set of all vertices and E = {$e_1, e_2, e_3, e_4, e_5, e_6, e_7$} be a set of edges which is relation on $V \times V$.



Fi

Let, we take a parameter set A ={ v_1, v_3, v_4}

Define set valued function F by,

F(x) = {y ∈ V| xRy⇔ $d(x,y) \leq 1$}.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

Then, F(v_1)={v_1, v_2, v_5}, F(v_3)={v_2, v_3, v_4}, F(v_4) ={v_2, v_3, v_4, v_5},

Here subgraph induced by F(x) in G, $F^{\%}$(x) is a connected subgraph of G, for all x∈A.

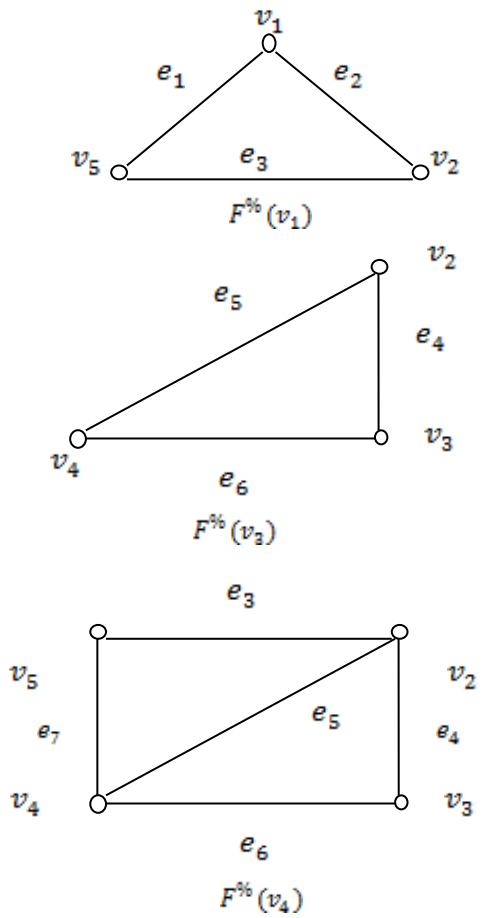


Fig – 2

Hence, (F,A) ∈SG(G) in Figure 2.

Definition 2.3

Let, G = (V, E) be the simple graph. A bethe parameter set and F be set valued function. (F, A) be a set of all soft graphs of G. The adjacency of matrix $M_{SG} = (a_{ij})$, where a_{ij} = 1 if there is an edge between vertex i and vertex j and a_{ij} = 0 otherwise. The matrix M_{SG} is called Soft Graph Matrix. Where, M_{SG} is set of all matrices of soft graphs in G.

Example 2.4

In Example 2.2, we saw about soft graphs. Here, we define the soft graph matrices for example 2.2.

Adjacency matrix for a graph G is,

$$M_G = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Soft graph matrices are defined as in the example 1

$$M_{SG[F^{\%}(v_1)]} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$M_{SG[F^{\%}(v_3)]} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$M_{SG[F^{\%}(v_4)]} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Definition 2.5

Let, G = (V, E) be the simple graph, A be a parameter set with a set valued function F. (F, A) be a set of all soft graphs of the G. M_{SG} be a set of all matrices of soft graphs of G. Let $M_{SG[F^{\%}(x_1)]} = m_{ij}$ and $M_{SG[F^{\%}(x_2)]} = n_{ij}$ are two soft graph matrices of G, where $x_1, x_2 \in A$. The Cartesian Product for two soft graph matrices is defined in the following way:

$M_{SG[F^{\%}(x_1)]} \times M_{SG[F^{\%}(x_2)]} = M_{SG[F^{\%}(x)]}$

Where, $M_{SG[F^{\%}(x)]}$ is also a soft graph matrix of same order that of $M_{SG[F^{\%}(x_1)]}$ or $M_{SG[F^{\%}(x_2)]}$ and each element is defined by $l_{ij} = m_{ij} \times n_{ij}$.

Definition 2.6

Let, M_{SG} be the set of all matrices of soft graphs of G. Let $M_{SG[F^{\%}(x_1)]} = m_{ij}$ and $M_{SG[F^{\%}(x_2)]} = n_{ij}$ are

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

two soft graph matrices of G, where $x_1, x_2 \in A$. Then, AND of two soft graph matrices is defined in the following way:

$$M_{SG[F^{\%}(x_1)]} \wedge M_{SG[F^{\%}(x_2)]} = M_{SG[F^{\%}(x)]}$$

Where, $M_{SG[F^{\%}(x)]}$ is also a soft graph matrix of same order that of $M_{SG[F^{\%}(x_1)]}$ *or* $M_{SG[F^{\%}(x_2)]}$ and each element is defined by $l_{ij} = m_{ij} \wedge n_{ij}$

Definition 2.7

Let, M_{SG} be the set of all matrices of soft graphs of G. Let $M_{SG[F^{\%}(x_1)]} = m_{ij}$ and $M_{SG[F^{\%}(x_2)]} = n_{ij}$ are two soft graph matrices of G, where $x_1, x_2 \in A$. Then, OR of two soft graph matrices is defined in the following way:

$$M_{SG[F^{\%}(x_1)]} \vee M_{SG[F^{\%}(x_2)]} = M_{SG[F^{\%}(x)]}$$

Where, $M_{SG[F^{\%}(x)]}$ is also a soft graph matrix of same order that of $M_{SG[F^{\%}(x_1)]}$ *or* $M_{SG[F^{\%}(x_2)]}$ and each element is defined by $l_{ij} = m_{ij} \vee n_{ij}$.

Definition 2.8

Let, M_{SG} be a set of all matrices of soft graphs of G. Let $M_{SG[F^{\%}(x)]} = m_{ij}$ be the soft graph matrix of G, where $x \in A$. Complement of a soft graph matrix is defined in the following way:

$$M_{SG[F^{\%}(x)]}{}^{c} = M_G - M_{SG[F^{\%}(x)]}$$

Where, $M_{SG[F^{\%}(x)]}{}^{c}$ is also a soft graph matrix of same order that of $M_{SG[F^{\%}(x)]}$.

3. MATLAB IMPLEMENTATION FOR SOFT GRAPH MATRIX

In this, a MATLAB program was developed to create soft graph matrices using a specified base graph and a set of parameters. Each parameter is linked to a subset of vertices, and the program generates matrices that depict the interconnections within these subsets. The dimensions of the original graph are preserved in all the matrices produced, with irrelevant entries being assigned a zero value. This technique offers a practical means of applying soft set theory to graphs in a computational setting.

Code:

```
%% Step 1: Define the base graph G (Adjacency
Matrix)

MG = [0 1 0 1 0;

    1 0 1 0 0;

 0 1 0 1 1;

    1 0 1 0 1;

 0 0 1 1 0];

%% Step 2: Define soft set mappings

F = struct();

F.a1 = [1 2 4];

F.a2 = [2 3 5];

F.a3 = [3 4 5];

parameters = fieldnames(F);

%% Step 3: Create subplot layout (2x2)

figure;

% --- Base Graph ---

subplot(2,2,1);

Gbase = graph(MG);

plot(Gbase, ...

    'Layout','force', ...

    'NodeLabel',1:5, ...

    'LineWidth',1.5, ...

    'MarkerSize',7, ...

    'NodeColor','b');              % Blue for
base graph

title('Base Graph (MG)');

% --- Soft Graphs ---

for i = 1:length(parameters)

paramName = parameters[4];

vertexSet = F.(paramName);

% Construct softMatrix for this parameter

softMatrix = zeros(5);
```

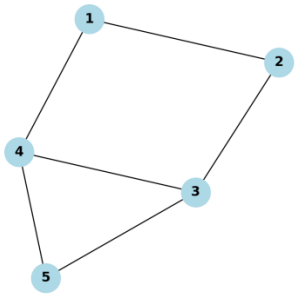
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

```
for r = 1:length(vertexSet)

for c = 1:length(vertexSet)

vr = vertexSet(r);

vc = vertexSet(c);

softMatrix(vr, vc) = MG(vr, vc);

end

end

    % Graph object

Gsoft = graph(softMatrix);

subplot(2,2,i+1);

plot(Gsoft, ...

        'Layout','force', ...

        'NodeLabel',1:5, ...

        'LineWidth',1.5, ...

        'MarkerSize',7, ...

        'NodeColor','r');

title(['Soft Graph ', paramName]);

end
```

Output:

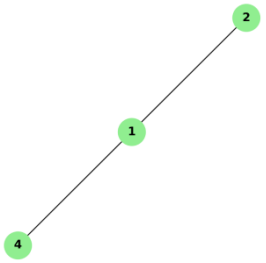
```
Base Graph Adjacency Matrix (MG):
     0     1     0     1     0
     1     0     1     0     0
     0     1     0     1     1
     1     0     1     0     1
     0     0     1     1     0
```

Base Graph (MG)

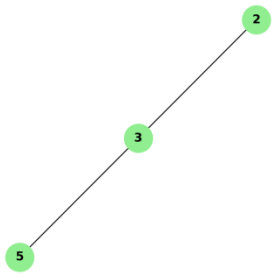



```
Soft Graph Matrices MSG[F%(x)]:

    MSG[F%(a1)] =
     0     1     0     1     0
     1     0     0     0     0
     0     0     0     0     0
     1     0     0     0     0
     0     0     0     0     0

    MSG[F%(a2)] =
     0     0     0     0     0
     0     0     1     0     0
     0     1     0     0     1
     0     0     0     0     0
     0     0     1     0     0

    MSG[F%(a3)] =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     1     1
     0     0     1     0     1
     0     0     1     1     0
```

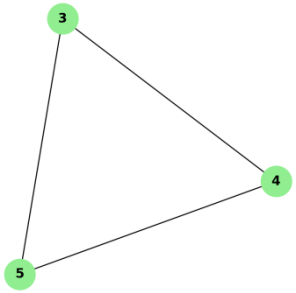
Soft Graph a1



Soft Graph a2



Soft Graph a3



International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

3.1 MATLAB Programming for Matrix Operations

3.1.1 Cartesian Product

Computes the **Cartesian product** of two soft graph matrices by performing **element-wise multiplication**.

Code:

```
functionresult = softMatrixCartesian(M1,
M2)
% Cartesian product (element-wise
multiplication)
result = M1 .* M2;
end
```

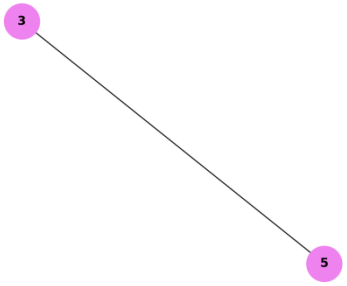
Example:

```
A1 = [0 1 0 1 0;
   1 0 0 0 0;
   0 0 0 0 0;
   1 0 0 0 0;
  0 0 0 0 0];

A2 = [0 0 0 0 0;
   0 0 1 0 0;
   0 1 0 0 1;
   0 0 0 0 0;
  0 0 1 0 0];
A3 = [ 0 0 0 0 0;
   0 0 0 0 0;
   0 0 0 1 1;
   0 0 1 0 1;
  0 0 1 1 0];

% Call Cartesian product function

cartesianResult = softMatrixCartesian(A2,
A3);
disp('Cartesian Product (A2, A3) =');
disp(cartesianResult);
% Plot Cartesian Product Graph
G = graph(cartesianResult);

figure;
plot(G, ...
    'Layout','force', ...
    'NodeLabel',1:numnodes(G), ...
    'LineWidth',1.5, ...
    'MarkerSize',7, ...
    'NodeColor','m');   % Magenta nodes
title('Cartesian Product Graph (A2 × A3)');
```

Output:

```
Cartesian Product (A2, A3) =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     1
     0     0     0     0     0
     0     0     1     0     0
```

```
Cartesian Product Graph (A2 × A3)
```



3.1.2 AND Operation

Performs **element-wise AND** operation between two soft graph matrices. For binary matrices, this is equivalent to taking the **minimum** of corresponding elements.

Code:

```
functionresult = softMatrixAND(M1, M2)
% Performs AND operation (element-wise
minimum)
result = min(M1, M2);
end
```

Example:

```
% Using A1 and A2 from above
A1 = [0 1 0 1 0;
   1 0 0 0 0;
   0 0 0 0 0;
   1 0 0 0 0;
  0 0 0 0 0];

A2 = [0 0 0 0 0;
   0 0 1 0 0;
   0 1 0 0 1;
   0 0 0 0 0;
  0 0 1 0 0];

% Call AND function

andResult = softMatrixAND(A1, A2);
disp('AND(A1, A2) =');
disp(andResult);
```

Output:

```
AND(A1, A2) =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

3.1.3 OR Operation

Performs **element-wise OR** operation between two soft graph matrices. This takes the **maximum** of each corresponding element.

Code:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

```
   functionresult = softMatrixOR(M1, M2)
% Performs OR operation (element-wise
maximum)
result = max(M1, M2);
end
```

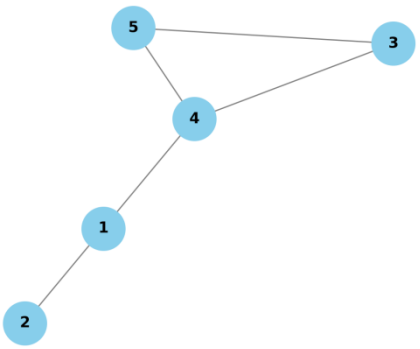
Example:

```
        % Using A1 and A3 from above
               A1 = [0 1 0 1 0;
                  1 0 0 0 0;
                  0 0 0 0 0;
                  1 0 0 0 0;
                 0 0 0 0 0];

               A3 = [0 0 0 0 0;
                  0 0 0 0 0;
                  0 0 0 1 1;
                  0 0 1 0 1;
                 0 0 1 1 0];

% Call OR function
orResult = softMatrixOR(A1, A3);
disp('OR(A1, A3) =');
disp(orResult);

% Plot the graph

G = graph(orResult);        % create graph
from adjacency matrix
figure;
plot(G, 'Layout', 'force'); % force-
directed layout
title('Graph of OR(A1, A3)');
```

Output:

```
OR(A1, A3) =
            0     1     0     1     0
            1     0     0     0     0
            0     0     0     1     1
            1     0     1     0     1
            0     0     1     1     0
```

Graph of OR(A1, A3)



3.1.4 Complement

Returns the **complement** of a binary soft graph matrix, replacing 1 with 0 and 0 with 1.

Code:

```
functionresult = softMatrixComplement(M)
% Complement of a binary matrix (1 - M)
result = 1 - M;
end
```

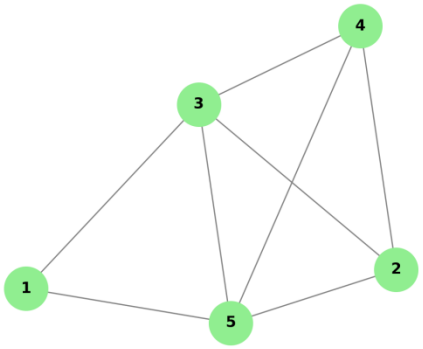
Example:

```
% Using A1 from above
               A1 = [0 1 0 1 0;
                  1 0 0 0 0;
                  0 0 0 0 0;
                  1 0 0 0 0;
                 0 0 0 0 0];

% Call complement function

complementResult =
softMatrixComplement(A1);
disp('Complement of A1 =');
disp(complementResult);
% Plot the graph

G = graph(complementResult);        %
create graph from adjacency matrix
figure;
plot(G, 'Layout', 'force'); % force-
directed layout
title('Graph of Complement of A1');
```

Output:

```
Complement of A1 =
            1     0     1     0     1
            0     1     1     1     1
            1     1     1     1     1
            0     1     1     1     1
            1     1     1     1     1
```

Graph of Complement of A1



4. APPLICATION

A textile manufacturer, who was located in the Coimbatore, distributes the products to international markets. To verify the quality, company runs 2 centers in Salem and

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

Karur and utilizes2 primary shipping ports - Chennai and Tuticorinto deliver to its overseas destinations: such as the Dubai, London, and New York. Every shipping route includes the specific investment costs and the anticipated profit margins. To increase the profitability when managing the uncertainties in cost and quality, it uses the soft graph theory. This model took 3 main parameters: a_1 (Low Cost), a_2 (High Quality), and a_3 (High Profit).

```
**Original Graph Nodes (V):**

1. Coimbatore
2. Salem
3. Karur
4. Chennai Port
5. Tuticorin Port
6. Dubai
7. London
8. New York

**Original Graph Edges (E) with Cost and Profit:**

* Coimbatore → Salem: Cost = 100
* Coimbatore → Karur: Cost = 90
* Salem → Chennai Port: Cost = 60
* Karur → Tuticorin Port: Cost = 70
* Chennai Port → Dubai: Profit = 500
* Tuticorin Port → London: Profit = 600
* Tuticorin Port → New York: Profit = 750


**Soft Parameters and the Corresponding Routes:**

* a1: Low Cost Route = Coimbatore → Karur → Tuticorin Port → London
* a2: High Quality Route = Coimbatore → Salem → Chennai Port → Dubai
* a3: High Profit Route = Coimbatore → Karur → Tuticorin Port → New York


**Soft Graph Matrices:**

**MSG1 (a1 - Low Cost):**

[ 0    0  -90   0     0     0    0     0;
   0    0    0   0     0     0    0
0;
   0    0    0   0   -70     0    0
0;
   0    0    0   0     0     0    0
0;
   0    0    0   0     0     0  +600
0;
   0    0    0   0     0     0    0
0;
   0    0    0   0     0     0    0
0;
   0    0    0   0     0     0    0     0
]

**MSG2 (a2 - High Quality):**

[ 0  -100   0     0     0     0    0
0;
   0    0    0   -60     0     0    0
0;
   0    0    0     0     0     0    0
0;
   0    0    0     0     0  +500    0
0;
   0    0    0     0     0     0    0
0;
   0    0    0     0     0     0    0
0;
   0    0    0     0     0     0    0
0;
   0    0    0     0     0     0    0
0 ]

**MSG3 (a3 - High Profit):**

[ 0    0  -90   0     0     0    0     0;
   0    0    0   0     0     0    0
0;
   0    0    0   0   -70     0    0
0;
   0    0    0   0     0     0    0
0;
   0    0    0   0     0     0    0
+750;
   0    0    0   0     0     0    0
0;
   0    0    0   0     0     0    0
0;
   0    0    0   0     0     0    0     0
]


MATLAB Code:

% Nodes
nodes =
{'Coimbatore','Salem','Karur','Chennai
Port','TuticorinPort','Dubai','London','N
ew York'};
n = length(nodes);

%% ---------- Base Graph (Original Graph)
----------
BaseG = zeros(n);

% Costs
BaseG(1,2) = -100;    % Coimbatore → Salem
BaseG(1,3) = -90;     % Coimbatore → Karur
BaseG(2,4) = -60;     % Salem → Chennai
Port
BaseG(3,5) = -70;     % Karur → Tuticorin
Port
```

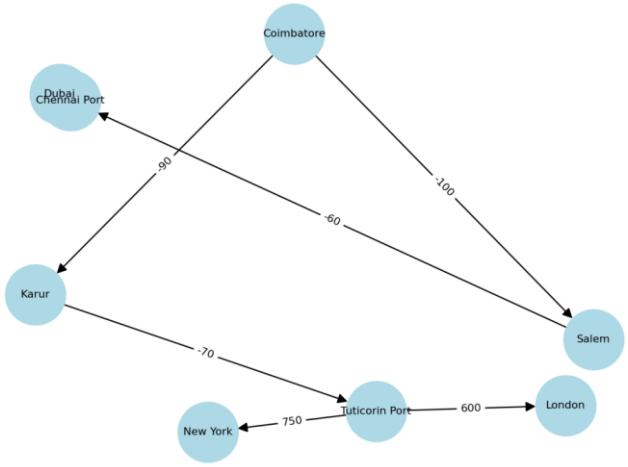
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

```
% Profits
BaseG(4,6) = 500;     % Chennai Port →
Dubai
BaseG(5,7) = 600;     % Tuticorin Port →
London
BaseG(5,8) = 750;     % Tuticorin Port →
New York

% ----- Plot Base Graph -----
figure;
G_base = digraph(BaseG, nodes);
p1 =
plot(G_base,'Layout','layered','EdgeLabel
',G_base.Edges.Weight);
title('Base Graph (Original Routes)');
p1.NodeColor = 'magenta';
p1.MarkerSize = 7;
p1.LineWidth = 1.5;
%% ---------- Soft Graph MSG1 (Low Cost
Route a1) ----------
MSG1 = zeros(n);
MSG1(1,3) = -90;     % Coimbatore → Karur
MSG1(3,5) = -70;     % Karur → Tuticorin
Port
MSG1(5,7) = 600;     % Tuticorin Port →
London

% ----- Plot MSG1 -----
figure;
G1 = digraph(MSG1, nodes);
p1 =
plot(G1,'Layout','layered','EdgeLabel',G1
.Edges.Weight);
title('Soft Graph MSG1 (Low Cost
Route)');
p1.NodeColor = 'red';
p1.MarkerSize = 7;
p1.LineWidth = 1.5;

%% ---------- Soft Graph MSG2 (High
Quality Route a2) ----------
MSG2 = zeros(n);
MSG2(1,2) = -100;    % Coimbatore → Salem
MSG2(2,4) = -60;     % Salem → Chennai
Port
MSG2(4,6) = 500;     % Chennai Port →
Dubai

% ----- Plot MSG2 -----
figure;
G2 = digraph(MSG2, nodes);
p2 =
plot(G2,'Layout','layered','EdgeLabel',G2
.Edges.Weight);
title('Soft Graph MSG2 (High Quality
Route)');
p2.NodeColor = 'cyan';
p2.MarkerSize = 7;
p2.LineWidth = 1.5;

%% ---------- Soft Graph MSG3 (High
Profit Route a3) ----------
MSG3 = zeros(n);
MSG3(1,3) = -90;     % Coimbatore → Karur
MSG3(3,5) = -70;     % Karur → Tuticorin
Port
MSG3(5,8) = 750;     % Tuticorin Port →
New York

% ----- Plot MSG3 -----
figure;
G3 = digraph(MSG3, nodes);
p3 =
plot(G3,'Layout','layered','EdgeLabel',G3
.Edges.Weight);
title('Soft Graph MSG3 (High Profit
Route)');
p3.NodeColor = 'green';
p3.MarkerSize = 7;
p3.LineWidth = 1.5;

%% ---------- OR Operation (MSG2 ∪ MSG3)
----------
MSG_OR = zeros(n);
for i = 1:n
for j = 1:n
if MSG2(i,j) == 0
            MSG_OR(i,j) = MSG3(i,j);
elseif MSG3(i,j) == 0
            MSG_OR(i,j) = MSG2(i,j);
else
if MSG2(i,j) > 0 && MSG3(i,j) > 0
                MSG_OR(i,j) =
max(MSG2(i,j), MSG3(i,j)); % max profit
else
                MSG_OR(i,j) =
min(MSG2(i,j), MSG3(i,j)); % min cost
end
end
end
end

% ----- Plot OR Graph -----
figure;
G_or = digraph(MSG_OR, nodes);
p4 =
plot(G_or,'Layout','layered','EdgeLabel',
G_or.Edges.Weight);
title('Soft Graph OR Operation (High
Quality ∪ High Profit)');
p4.NodeColor = 'yellow';
p4.MarkerSize = 7;
p4.LineWidth = 1.5;

%% ---------- Net Profit Calculation ----
------
net_profit = sum(MSG_OR(MSG_OR ~= 0));
fprintf('Net Profit (High Quality ∪ High
Profit): ₹%d\n', net_profit);
```

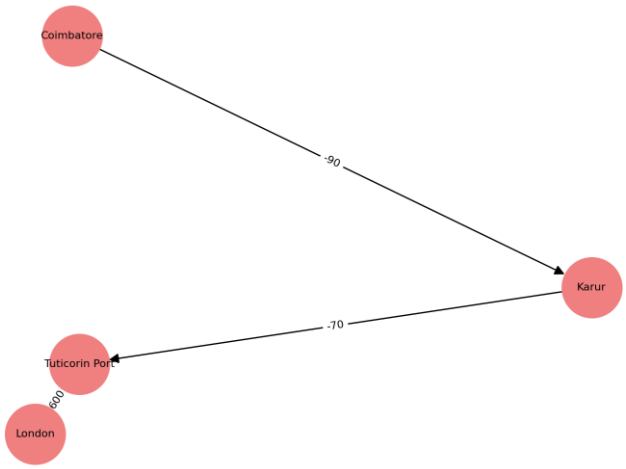
Output:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

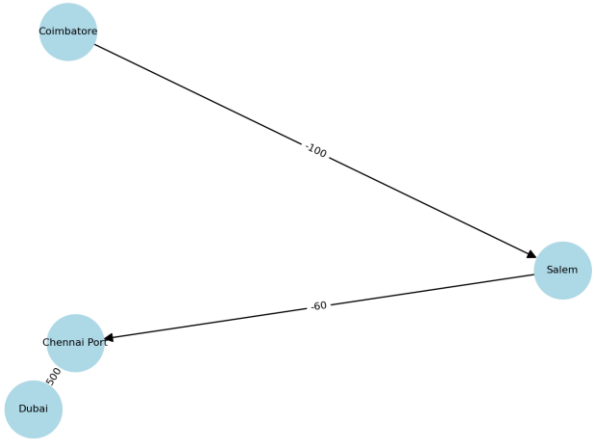
Base Graph



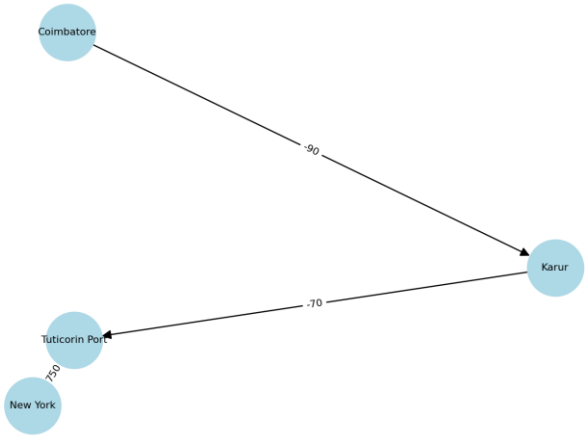
Soft Graph MSG1 (Low Cost Route)



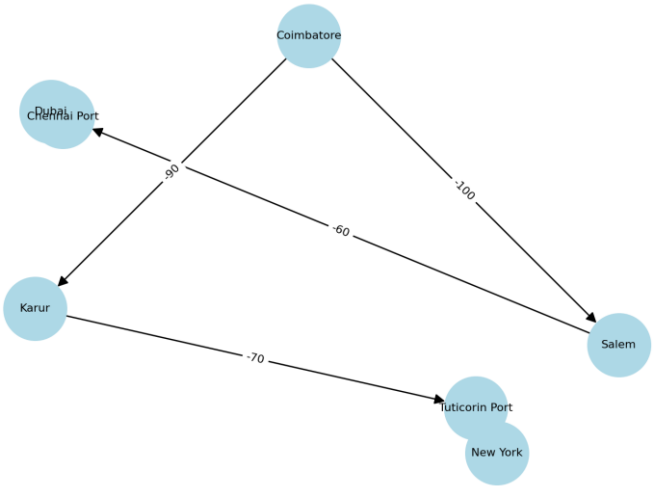
Soft Graph MSG2 (High Quality Route)



Soft Graph MSG3 (High Profit Route)



Soft Graph OR Operation (High Quality U High Profit)



Net Profit (High Quality U High Profit): 930

The case study highlightedsoft graph matrices role in enabling the decision-making when faced with the uncertainty. By analyzing the parameters such as the cost, quality, and profit, MATLAB implementation identifies optimal trade route, which yields the net profit of Rs. 930. This demonstrated practical application of the soft graph theory by solving the real-world supply chain and the logistics issues.

5. CONCLUSION

This paper presented the effective and simple MATLAB framework for working with soft graph matrices, and it is built on soft set theory & graph theory. Also, it creates the matrices from the base graph with given parameters and then supports the key operations such as the AND, OR, Cartesian product, and complement. The full-matrix method keeps the analysis consistent when leaving room for upgrades. The work links the theory with practice, useful in uncertain environments, and it can be extended with the user-friendly tools, real-time scalability, and applications in decision-making, social networks, and recommendations. The Future scope include fuzzy or hybrid soft graphs for the more complex uses.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

6. REFERENCES

[1] D. Molodtsov, "Soft set theory—first results," *Computers & mathematics with applications,* vol. 37, no. 4-5, pp. 19-31, 1999.
[2] R. K Thumbakara and B. George, "Soft graphs," *General Mathematics Notes,* vol. 21, pp. 75-86, 01/01 2014.
[3] L. Euler, "Solutio problematis ad geometriam situs pertinentis," *Commentarii academiae scientiarum Petropolitanae,* pp. 128-140, 1741.
[4] N. Biggs, *Algebraic graph theory* (no. 67). Cambridge university press, 1993.
[5] T. Boffey, "Graph theory in operations research," *(No Title),* 1982.
[6] N. Deo, *Graph theory with applications to engineering and computer science*. Courier Dover Publications, 2016.
[7] J. L. Gross and T. W. Tucker, *Topological graph theory*. Courier Corporation, 2001.
[8] M.G. Karunambigai and V. Gnaneswaran, "Matrix Representation of Soft Graphs," 2020.