Sayyed Jalil, 2025, 13:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Graph Based decision making method using Soft Set Relations

[1]Sayyed Jalil, [2]Rahul Deshmukh

[1]Assistant Professor, Department of Mathematics,
Hutatma Jaywantrao Patil College, Himayatnagar,
Maharashtra 431802, India, drsayyedjalil@gmail.com,
ORCID:0009-0003-2152-505X

[2]Research Scholar, School of Mathematical Sciences,
Swami Ramanand Teerth Marathwada University,
Nanded, 431606, India, deshmukh2776@gmail.com,
ORCID:0009-0001-7192-1806

Abstract: In this paper, we describe a novel approach to analyzing and evaluating complex choice issues by utilizing graph representations of soft set relations in our choice-making method. By looking at the incoming and outgoing values of choices inside a graph-based framework, the suggested algorithm offers an organized method for determining the optimum option. This method may be adapted to a variety of decision-making models and is especially helpful for tackling issues in a variety of domains, including social decision-making**.**

Keywords: Soft Set, Decision Making Problem, Representation of Soft Set Relations, Directed graph.

1. Introduction

Soft Set Theory was introduced by D. Molodtsov [6] and provides a framework for dealing with uncertainties and imprecise data, which is crucial in various fields such as engineering, social sciences, and economics. The theory has been successfully applied in areas like game theory and operation research, highlighting its versatility and potential for practical applications. Unlike fuzzy set theory, rough set theory, or other approaches to vagueness, soft set theory relies on parametrization. This means that the uncertainty is handled by associating attributes or characteristics to the objects under consideration. Soft set theory has found applications in diverse fields, including decision-making, medical diagnosis, pattern recognition, and engineering [9]. A graph's representation can greatly benefit by knowing each vertex's neighborhood. A new tool for such representation is offered by soft set theory. Authors Ali et al.[2] presented a graph representation method based on soft set theory and adjacency of vertices. Many new facets of graph theory may be revealed by applying algebraic operations that are available in soft sets to this representation of a graph. Furthermore, a metric is established to determine the separations between graphs that are represented by soft sets. B. Reddy et al.[10] presents a software and algorithm for attribute reduction using multiple soft sets. Applications of the program to farmers' crop selection are discussed. Polat et al.[8] introduces a decision-making method using a graph representation of soft set relations. The approach uses directed graphs to visualize soft-set relations, offering a new tool for solving decision-making problems. The method's applicability is demonstrated through examples. In 2024, the literature survey on soft sets was done by J.Alcantud, A.Z.Khameneh, and G. Santos-Garcıa[1]. Zhang[15] defined interval soft sets and used them to solve a decision-making problem in 2014. In his paper, interval soft sets are represented by a table from which, he derived an interval choice value.

Soft set theory naturally leads to soft matrix theory, which offers a practical means of representing and working with soft sets. A matrix with soft sets as its entries is referred to as a soft matrix in this context.

These matrices make analysis and calculation easier by tabulating the relationships between parameters and objects. The authors N. Cagman and S. Enginoglu[4] introduce the number of operations that have been described on soft matrices, such as soft max-min composition, which is especially helpful in ambiguous decision-making problems. Soft graphs arise from the combination of soft set theory and graph theory. A soft graph is a graph where the vertices or edges (or both) are defined by soft sets. This allows for the representation of graphs with uncertain or vague connections between nodes. Bobin George and R. K. Thumbakara [14] have provided a concept of the soft graph, which is important to this subject. Thenge et al.[12] introduces the notion of adjacency and incidence matrices for soft graphs. It explores how some classical results from graph theory may not hold true in soft graph theory. J. Thenge, B. Reddy, and R. Jain[11, 13] contribute to the development of soft graph theory by providing tools to analyze the structural properties of soft graphs using matrix representations. The study "Connected Soft Graph" [12] defines the concept of connected soft graphs and derives related results provided by J. Thenge et al. We have used some literature from J.A.Bondy and U.S.R. Murty[3] elementary definitions in the graph theory such as a graph, subgraph, etc.

2. Preliminaries

The idea of soft sets, first put forth by Molodtsov [6], is presented in this section. Several helpful terminologies about soft sets, graph related properties are also included. Here, U, E, and $P(U)$ represent the universal set, set of parameters, and power set respectively.

Definition 2.1 (Soft Set(Molodtsov)[6]). An ordered pair (S, X) is referred to as a soft set over U, where S is a function given by $S: X \to P(U)$. Thus, a soft set over U is a parameterized collection of subsets of the universal set U. A soft set is not the same as a crisp set.

Definition 2.2 (Soft subset(Molodtsov)[6]). For two soft sets (S_1, X) and (S_2, Y) over a common universe $U, (S_1, X)$ is a soft subset of (S_2, Y) if i.) $X \subset Y$ *and* ii.) For all $x \in X, S_1(x)$ and $S_2(x)$ represent the same approximation. We can write $(S_1, X) \subset (S_2, Y)$. (S_1, X) is said to be a soft superset of (S_2, Y), if (S_2, Y) is a soft subset of (S_1, X).

Definition 2.3 (Cartesian product(Molodtsov)[6]). Let (S, X) and (T, Y) be soft sets over a common universe U, then the Cartesian product of (S, X) and (T, Y) is defined as $(S, X) \times (T, Y) = (R, X \times Y)$, where $R: X \times Y \to P(U \times U)$ and $R(x, y) = S(x) \times T(y)$, where $(x, x) \in X \times Y$.

Definition 2.4 (Directed graph (Bondy & Murty)[3]). A directed graph $G = (V, E)$, or digraph, consists of a set V of vertices (or nodes) together with a set E of edges (or arcs). The vertex $a \in V$ is called the initial vertex of the edge $(a, b) \in E$, while the vertex $b \in V$ is called the terminal vertex of this edge. The edge (a, a) is called a loop. When a graph $G = (V, E)$ contains no loop, it is called loop-free. There is a path starting at $a \in V$ and ending at $b \in V (a \neq b)$. Such a path consists of a finite sequence of directed edges.

3. Result and Discussion

Consider the following example.

Example 3.1. Let U be a set of all students under consideration. X is a set of parameters. Each parameter can be a word or sentence. $X =$ { brilliant, average, healthy }. In this case, we can define a soft set (S, X) to point out the Nature of students as follows:

Suppose that there are six students in the universe U given by $U = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ and $X = \{e_1, e_2, e_3\}$ where e_1 stands for brilliant, e_2 stands for average and e_3 stands for healthy.

The soft set (S, X) defined as $S(e_1) = \{x_1, x_2, x_6\}, S(e_2) = \{x_3, x_4, x_6\}, S(e_3) = \{x_1, x_4, x_5, x_6\}$ gives the soft set representing the nature of students.

The soft set (S, X) is a parametrized family. $\{S(e_i): i = 1,2,3\}$ of subsets of the set U and gives us a collection of approximate descriptions of an object. So the soft set (S, X) is called a standard soft set.

To store soft sets in computers, one may need the representation of soft sets in the form of datasets. The (i, j)th entry in the table

$$A_{ij} = \begin{cases} 1 \text{ if } x_i \in F(e_j) \\ 0 \text{ otherwise} \end{cases}$$

Regarding the above, the tabular representation of the soft set is given in Table 1.

$U\downarrow$	e_1 (brilliant)	e_2 (average)	e_3 (healthy)
x_1	1	0	1
x_2	1	0	0
x_3	0	1	0
x_4	0	1	1
x_5	1	0	1
x_6	0	1	1

Table 1: Tabular Representation of the soft set

3.1 Directed graph of Soft Set Relation

Example 3.2. Let (S,X) be a soft set over U where $U=\{h_1,h_2,h_3,h_4,h_5,h_6\}, X=\{e_1,e_2,e_3,e_4,e_5\}$ and
$S(e_1)=\{h_1,h_2,h_4\}, S(e_2)=\{h_3,h_4\}, S(e_3)=\{h_2,h_4\},$
$S(e_4)=\{h_1,h_5\}, S(e_5)=\{h_1,h_4,h_6\}$

Let the soft set relation R on (S,X) is given as:

$$R=\{S(e_1)\times S(e_1), S(e_1)\times S(e_2), S(e_2)\times S(e_2), S(e_2)\times S(e_3), S(e_3)\times S(e_3), S(e_4)\times S(e_4), S(e_4)\times S(e_1), S(e_5)\times S(e_2)\}$$

We can represent R with a directed graph, named. G_R, as follows:

If we consider the set of vertices V and set of edges E as:

$V=\{S(e_1),S(e_2),S(e_3),S(e_4)\}$
and
$E=\{(S(e_1),S(e_1)),(S(e_1),S(e_2)),S(e_2),S(e_2)),(S(e_2),S(e_3))$
,
$(S(e_3),S(e_3)),(S(e_4),S(e_4)),(S(e_4),S(e_1)),(S(e_5),S(e_2))\}$
respectively then the graph $G_R=(V,E)$ will be the Graph Representation of R.

An edge of the form $(S(e_1),S(e_1))$ is illustrated by are from the vertex $S(e_1)$ back to itself. Such an edge is called a loop.

The directed graph G_R for the soft set relation, R can be illustrated in Figure 1 below:

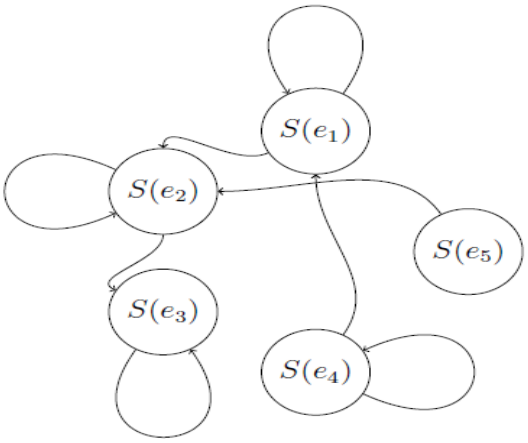

Figure 1: Directed graph G_R

3.2 A method for decision-making problems using the graph representation of soft set relation

Our method is discussed below with an algorithm and example:

3.2.1 Algorithm

Input:

- $U=\{b_1,b_2,b_3,b_4,b_5,b_6\}$: Universal set of beaches.
- $X=\{e_1,e_2,e_3,e_4,e_5\}$: Parameter set (attractiveness factors).
- $S: X\to P(U)$: Soft set representing the mapping of each attractiveness factor to the subset of beaches.
- Graph representation showing incoming and outgoing paths between the attractiveness parameters.

Step 1:

Count Incoming and Outgoing Paths:
For each vertex e_i, count the number of incoming paths and outgoing paths. Represent each vertex with an ordered pair (incoming, outgoing).

Example for Vertex e_1 :
Incoming paths: Number of arrows directed toward e_1.
Outgoing paths: Number of arrows directed away from e_1.

Step 2:

Vector Representation of Soft Set $S(e_i)$:
Convert the image of each e_i (the subset of beaches associated with e_i) into a vector representing the existence of beaches in the universal set U. Each component in the vector

corresponds to a beach, where 1 indicates the beach is present in $S(e_i)$ and 0 indicates the beach is not present.

Step 3:

Adjust the Incoming and Outgoing Values for Each Vector:

For each attractiveness parameter e_i, adjust the vector representation by adding the number of incoming and outgoing paths to each component of the vector.

Step 4:

Compute Total Incoming and Outgoing Values for Each Beach b_i :

For each beach b_i, sum the adjusted values from each attractiveness vector across all e_i.

Step 5:

Select the Optimal Beach :

Maximal Incoming Value: Choose the beach with the highest incoming value.

Tie-breaking: If multiple beaches have the same maximal incoming value, choose the one with the highest outgoing value among them.

This algorithm provides a systematic way to rank and choose the most attractive beach based on the parameters defined in the soft set framework.

Example 3.3. A soft set (S, X) consists of a mapping $S: X \to P(U)$ where $P(U)$ is the power set of U, the universal set of beaches. Each parameter $e_i \in X$ is associated with a subset of U, which contains the beaches that satisfy that particular parameter.
Let's define the soft set (S, X) where the attractiveness of each beach b_i in the universal set, U is described according to the parameters in X :
U={b_1= Tarkarli Beach (Sindhudurg),
b_2= Alibaug Beach (Raigad),
b_3=Ganpatipule Beach (Ratnagiri)
b_4=Bhogave beach(Sindhudurg)
b_5= Vengurla-Sagareshwar beach(Sindhudurg)
b_6= Baga beach (Goa)}

$X = \{e_1 =$ beautiful, $e_2 =$ clean, $e_3 =$ lonely,$e_4 =$ smooth and sandy,$e_5 =$ breezy $\}$

Now, we define the soft set S :

$S(e_1) = \{b_1, b_2, b_3, b_5\}$ (beaches considered beautiful)
$S(e_2) = \{b_1, b_3, b_4, b_6\}$ (beaches considered clean)
$S(e_3) = \{b_1, b_3, b_5\}$ (beaches considered lonely)
$S(e_4) = \{b_2, b_3, b_4\}$ (beaches smooth & sandy terrain)
$S(e_5) = \{b_1, b_4, b_5, b_6\}$ (breezy beaches)

Thus, the soft set (S, X) describes the attractiveness of the beaches in the Konkan and Goa regions according to the specified parameters. Each parameter e_i maps to a subset of beaches $b_i \in U$ that exhibit that specific attribute.

Let Mr. Z's priorities be ranked as beautiful, smooth and sandy, lonely, breezy, and clean beaches. According to this priority ranking, we can define a soft set relation on (S, X) as follows:

$$S(e_1) = \{b_1, b_3\}, S(e_2) = \{b_2, b_3, b_6\}, S(e_3) = \{b_4, b_5\},$$
$$S(e_4) = \{b_1, b_4\}, S(e_5) = \{b_2\}$$
$$R = \{S(e_3) \times S(e_1), S(e_3) \times S(e_4), S(e_3) \times S(e_5), S(e_2) \times S(e_3), S(e_1) \times S(e_4), S(e_1) \times S(e_5), S(e_1) \times S(e_2), S(e_4) \times S(e_5), S(e_4) \times S(e_2), S(e_5) \times S(e_2)\}$$

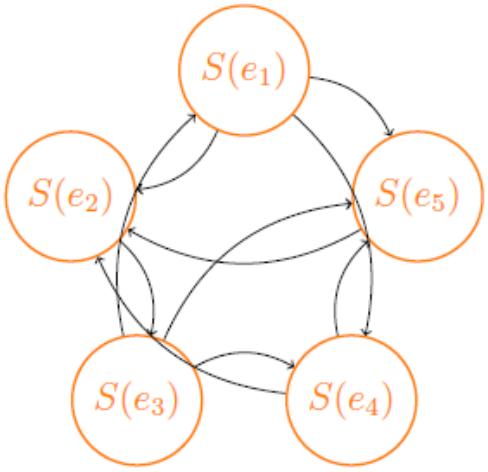

Figure 2 : Directed Graph

Step 1:

Each vertex's entering and exiting paths should be counted and represented using ordered pairs, where the number of incoming ways is the first component and the number of departing ways is the second.

$S(e_1)$	(1,3)
$S(e_2)$	(3,1)
$S(e_3)$	(1,3)
$S(e_4)$	(2,2)
$S(e_6^-)$	(3,1)

Table 2: Order Pairs

For the node $S(e_1)$ in the above example, we have 1 incoming and 3 outgoing ways. Thus, we get the pair $(1,3)$ for the node $S(e_1)$. In this example, all ways are listed in table 2:

Step 2:
Examine the images of each $S(e_i)$ to determine the existence of the elements of a universal set. For this purpose, we will see the images $S(e_i)$ as vectors and each of its elements represents the existence of elements of the universal set in the same order. After applying the method to the above example, the existence vectors are as follows:

$(1,1,1,0,1,0)$ for $S(e_1) = \{b_1, b_2, b_3, b_5\}$
$(1,0,1,1,0,1)$ for $S(e_2) = \{b_1, b_3, b_4, b_6\}$
$(1,0,1,0,1,0)$ for $S(e_3) = \{b_1, b_3, b_5\}$
$(0,1,1,1,0,0)$ for $S(e_4) = \{b_2, b_3, b_4\}$
$(1,0,0,1,1,1)$ for $S(e_5) = \{b_1, b_4, b_5, b_6\}$

Step 3:
Find the incoming & outgoing values for each $S(e_i)$. We can find the values by adding the numbers of incoming & outgoing ways of $S(e_1)$ to each component of vectors. Incoming & outgoing values are evaluated below as follows. The first vectors represent incoming values while the second represents outgoing values for b_1, b_2, b_3, b_4, b_5 and b_6 in $S(e_1), S(e_2), S(e_3), S(e_4)$, and $S(e_5)$ respectively:

$S(e_1)$	(2,2,2,0,2,0)	(4,4,4,0,4,0)
$S(e_2)$	(4,0,4,4,0,4)	(2,0,2,2,0,2)
$S(e_3)$	(2,0,2,0,2,0)	(4,0,4,0,4,0)
$S(e_4)$	(0,3,3,3,0,0)	(0,3,3,3,0,0)
$S(e_5)$	(4,0,0,4,4,4)	(2,0,0,2,2,2)

Table 3: Representation

Step 4:
For every b_i, calculate the sum of the incoming and exiting values. We have a choice to the Maximum Incoming Value. If any items have maximal incoming values, we will be given the option based on the maximum departing value of those things.
Total incoming and outgoing values for b_1, b_2, b_3, b_4, b_5 and b_6 are
$(12,12), (5,7), (11,13), (11,7), (8,10)$ and $(8,4)$ respectively.

Step 5:
Hence the optimal beach will be b_1 = Tarkarli Beach (Sindhudurg) according to our algorithm

3.3 Python Implementation of the Algorithm for Optimal Beach Selection

```
import numpy as np
def beach_selection_decision():
    # Step 1: Define the universal set of beaches with their full names
    U = [
        "Tarkarli Beach (Sindhudurg)",
        "Alibaug Beach (Raigad)",
        "Ganpatipule Beach (Ratnagiri)",
        "Bhogave Beach (Sindhudurg)",
        "Vengurla-Sagareshwar Beach (Sindhudurg)",
        "Baga Beach (Goa)"
    ]
    # Define the parameter set (attractiveness factors)
    X = [
        "beautiful",
        "clean",
        "lonely",
        "smooth and sandy",
        "breezy"
    ]
    # Define the soft set mapping (which beaches have which attractiveness factors)
    S = {
        "beautiful": [0, 1, 2, 4],  # b1, b2, b3, b5
        "clean": [0, 2, 3, 5],      # b1, b3, b4, b6
        "lonely": [0, 2, 4],        # b1, b3, b5
        "smooth and sandy": [1, 2, 3], # b2, b3, b4
        "breezy": [0, 3, 4, 5]      # b1, b4, b5, b6
    }
    # Define the graph relations between parameters based on Mr. Z's priorities
    graph_relations = {
        "lonely": ["beautiful", "smooth and sandy", "breezy"],
        "clean": ["lonely"],
        "beautiful": ["smooth and sandy", "breezy", "clean"],
        "smooth and sandy": ["breezy", "clean"],
        "breezy": ["clean"]
    }
    # Step 1: Count incoming and outgoing paths for each parameter
    param_stats = {}
    for param in X:
        incoming = 0
        outgoing = 0
        # Count incoming paths (other parameters pointing to this one)
```

```
        for src, targets in graph_relations.items():
            if param in targets:
                incoming += 1
        # Count outgoing paths (this parameter pointing to others)
        if param in graph_relations:
            outgoing = len(graph_relations[param])
        param_stats[param] = (incoming, outgoing)
    print("Step 1 - Parameter statistics (incoming, outgoing):")
    for param, stats in param_stats.items():
        print("{0}:{1}".format(param,stats))
    print()

# Step 2: Create vector representations for each parameter's soft set
    param_vectors = {}
    for param in X:
        vector = np.zeros(len(U), dtype=int)
        for beach_idx in S[param]:
            vector[beach_idx] = 1
        param_vectors[param] = vector
    print("Step 2 - Vector representations:")
    for param, vector in param_vectors.items():
        print("{0}:{1}".format(param,vector))
    print()
    # Step 3: Adjust vectors with incoming and outgoing values
    adjusted_vectors_incoming = {}
    adjusted_vectors_outgoing = {}
    for param in X:
        incoming, outgoing = param_stats[param]
        vector = param_vectors[param]
        # Create adjusted vectors
        adj_incoming = vector * (1 + incoming)  # Add incoming count to existing 1s
        adj_outgoing = vector * (1 + outgoing)  # Add outgoing count to existing 1s
        adjusted_vectors_incoming[param] = adj_incoming
        adjusted_vectors_outgoing[param] = adj_outgoing
    print("Step 3 - Adjusted vectors (Incoming, Outgoing):")
    for param in X:
        print("{0}".format(param))
        print("Incoming: {0}".format(adjusted_vectors_incoming[param])
        print("Outgoing: {0}".format(adjusted_vectors_outgoing[param])
        print()
    # Step 4: Compute total incoming and outgoing for each beach
    total_incoming = np.zeros(len(U), dtype=int)
    total_outgoing = np.zeros(len(U), dtype=int)
    for param in X:
        total_incoming += adjusted_vectors_incoming[param]
        total_outgoing += adjusted_vectors_outgoing[param]
    print("Step 4 - Beach scores (Incoming, Outgoing):")
    for i in range(len(U)):
        print("{0}: Incoming={0}, Outgoing={0}".format(U[i], total_incoming[i], total_outgoing[i])
    print()
    # Step 5: Select an optimal beach
    max_incoming = max(total_incoming)
    candidates = [i for i, score in enumerate(total_incoming) if score == max_incoming]

     if len(candidates) == 1:
        optimal_idx = candidates[0]
    else:
        # In case of a tie, select the one with the highest outgoing score
        max_outgoing = -1
        optimal_idx = candidates[0]
        for idx in candidates:
            if total_outgoing[idx] > max_outgoing:
                max_outgoing = total_outgoing[idx]
                optimal_idx = idx
    print("Step 5 - Optimal beach selection:")
    print("The optimal beach is: {0}".format(U[optimal_idx]))
    print("With incoming score: {0} and outgoing score: {0}.format{ total_incoming[optimal_idx], total_outgoing[optimal_idx])
    return U[optimal_idx]
# Run the algorithm
optimal_beach = beach_selection_decision()
```

Output

```
Step 1 - Parameter statistics (incoming, outgoing):
beautiful: (1, 3)
clean: (3, 1)
lonely: (1, 3)
smooth and sandy: (2, 2)
breezy: (3, 1)
```

Step 2 - Vector representations:
beautiful: [1 1 1 0 1 0]
clean: [1 0 1 1 0 1]
lonely: [1 0 1 0 1 0]
smooth and sandy: [0 1 1 1 0 0]
breezy: [1 0 0 1 1 1]

Step 3 - Adjusted vectors (Incoming, Outgoing):
beautiful:
Incoming: [2 2 2 0 2 0]
Outgoing: [4 4 4 0 4 0]
clean:
Incoming: [4 0 4 4 0 4]
Outgoing: [2 0 2 2 0 2]
lonely:
Incoming: [2 0 2 0 2 0]
Outgoing: [4 0 4 0 4 0]
smooth and sandy:
Incoming: [0 3 3 3 0 0]
Outgoing: [0 3 3 3 0 0]
breezy:
Incoming: [4 0 0 4 4 4]
Outgoing: [2 0 0 2 2 2]

Step 4 - Beach scores (Incoming, Outgoing):
Tarkarli Beach (Sindhudurg): Incoming=12, Outgoing=12
Alibaug Beach (Raigad): Incoming=5, Outgoing=7
Ganpatipule Beach (Ratnagiri): Incoming=11, Outgoing=13
Bhogave Beach (Sindhudurg): Incoming=11, Outgoing=7
Vengurla-Sagareshwar Beach (Sindhudurg): Incoming=8, Outgoing=10
Baga Beach (Goa): Incoming=8, Outgoing=4

Step 5 - Optimal beach selection:
The optimal beach is: Tarkarli Beach (Sindhudurg)
With incoming score: 12 and outgoing score: 12

4. Conclusion

We provided a method with the program for analyzing soft set relations in this work by creating a Decision-Making Method based on a graph representation of these interactions. The above-summarized algorithm offers a methodical way to choose the best option based on soft-set relations. It evaluates the attractiveness of options by computing incoming and outgoing values using graph-based analysis. This method can be used in a variety of decision-making situations. To increase the algorithm's efficiency and scalability across a wider range of industries, future work may involve improving it to handle bigger, more complicated datasets, investigating its use in real-time decision systems, and further integrating it with machine learning and AI-based decision-making frameworks.

References

[1] J. Alcantud, A. Khameneh, G. Santos, A systematic literature review of soft set theory. Neural Computer & Applications 36(2024), pp 8951–8975, https://doi.org/10.1007/s00521-024-09552-x.

[2] M. Ali, M. Shabir, and F. Feng, Representation of graphs based on neighbourhoods and soft sets, Int. J. Mach. Learn. & Cyber. 8(2017), 1525–1535, https://doi.org/10.1007/s13042-016-0525-z.

[3] J. Bondy and U.Murty, Graph Theory with Applications, The· Macmillan Press Ltd, 1976.

[4] N. Cagman and S. Enginoglu, Soft matrix theory and its decision making, Computers and Mathematics with Applications, 59(2010), pp 3308-3314.

[5] P. Maji, R. Biswas and A. Roy, Soft set theory, Computers and Mathematics With Applications, 45(2003), 555–562.

[6] D. Molodtsov, Soft Set Theory First Results, Computers and Mathematics With Applications, 37(1999), 19–31.

[7] A. Muayad, A.J. Ameer, Al-swidi and A.A.Omran, Study of some graph types via soft graphs Journal of Engineering And Applied Sciences, 14(2019),no. 8, 10375–10379.

[8] N. Polat, G. Yaylali, B. Tanay, A Method for Decision Making Problems by Using Graph Representation of Soft Set Relations, Intelligent Automation and Soft Computing, 25(2019), no. 2, pp 305-–311, https://doi.org/10.31209/2018.100000006.

[9] K. Ratheesh, On Soft Graphs and Chained Soft Graphs, International Journal of Fuzzy System Applications (IJFSA) 2018 7(2).

[10] B. Reddy, S. Jalil and M. Hodeish, Programming and algorithm for attribute reduction using multi soft sets and its applications to crop selection, Open Journal of Applied and Theoretical Mathematics (OJATM), 2(2016), 1015–1026.

[11] J. Thenge, B. S. Reddy and R.S. Jain, Adjacency and Incidence Matrix of a Soft Graph, Communications in Mathematics and Applications,11(2020), no. 1,pp 23–30.

[12] J. Thenge, B. S. Reddy and R. S. Jain, Connected Soft Graph, New Mathematics and Natural Computation 16(2020), no. 2, 305–318, https://doi.org/10.1142/S1793005720500180.

[13] J. Thenge-Mashale, B.S. Reddy and R.S. Jain, Application of a Soft graph in decision making using adjacency matrix of soft graph, Ganita, 71(2021), no. 2, 181–188.

[14] R. Thumbakara and B. George, Soft Graphs, Gen. Math. Notes, 21(2014), no.2, 75–86.

[15] X Zhang, On Interval Soft Sets with Applications, International Journal of Computational Intelligence Systems, 7(1),(2014) pp 186-196, https://10.1080/18756891.2013.862354.