NSS

Neutrosophic Sets and Systems, Vol. 60, 2023

University of New Mexico

Computation of Neutrosophic Soft Topology using Python

J J Mershia Rabuni[1] * and N. Balamani[2]

[1,2]Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore - 641 043

*Correspondence: mershiarabuni@gmail.com

Abstract. While other programming languages are either losing ground or stagnant, Python's popularity is growing. Soft sets lack the ability to eliminate the uncertainties present in conventional approaches, whereas neutrosophic sets are capable of handling confusing and contradictory data. In order to reduce the number of human computations performed to compute union, intersection and complement of neutrosophic soft sets, programs are developed in Python. Additionally, Python is used to compute Neutrosophic soft topological operators like interior and closure. The developed python programs are documented in this paper.

Keywords: neutrosophic soft; topological operators; python; topology; open set

1. Introduction

Programming languages play a vital role in solving mathematical problems. Over the last decade, constructing codes using a programming language for mathematical problems has undergone unexpected growth. In earlier times, rigid computer programs could only solve limited-size models. But today, they consist of highly developed and adaptable information processing systems with modules that manage incoming data, provide powerful computational algorithms and present the model's results in a manner that is acceptable to an application-oriented user. Even basic mathematical problems need an unreasonably high amount of computational effort if solved manually. Computational programs make these calculations easier by providing the results in no time. Python is currently one of the most popular programming languages among developers and tech companies. Python is highly preferable for its simplicity.

Zadeh [29] developed the fuzzy set ($\mathcal{FS}$) theory in 1965. It has developed into a very important tool for addressing problems with uncertainity. Molodtsov [18] presented the soft set theory in 1991, which addresses uncertainty. In his work, he established the core ideas of this novel theory and successfully applied it to a number of areas, including optimisation, algebraic structures, operations research, clustering, game theory, medical diagnosis, lattice,

topology, data analysis, and decision-making under uncertainty. The notion of intuitionistic fuzzy set was developed by Atanassov [1] by generalizing the fuzzy set theory. Intuitionistic fuzzy set contains both belongingness and non-belongingness values. But it is incapable of handling the ambiguity and contradictory data present in any system.

Smarandache [25,26] presented the novel idea of a neutrosophic set ($\mathcal{NS}$) as a generalisation of crisp sets, fuzzy set theory, and intuitionistic fuzzy set theory. The novel branch of philosophy known as neurosophy generalises fuzzy logic, intuitionistic fuzzy logic, and paraconsistent logic. Neutrosophic logic serves as a mathematical toolbox for issues involving inconsistent, incomplete and ambiguous knowledge. Neutrosophic logic and sets are used in many different areas, including financial dataset detection, investigation of the rise and collapse of the new economy,relational database systems, semantic web services, information systems, etc.

A theoretical framework for soft set theory was constructed by Maji et al. [16] in 2003. They also constructed operations like union (OR) and intersection (AND) of two soft sets as well as put forth certain hypotheses regarding these operations. Maji et al. initiated the study of $\mathcal{F}_{\mathcal{S}}$ sets in [13]. The idea of IF_S sets was presented by Maji et al. [14], who also defined new operations on it and examined some of their characteristics. By combining the ideas of soft set and neutrosophic set, Maji [17] initiated the study on neutrosophic soft sets ($\mathcal{N}_{\mathcal{S}}\mathcal{S}$s) and presented a solution for a decision-making problem utilizing the neutrosophic soft set, which was later modified by Deli and Broumi [9].

C.L. Chang [5] was the first to put forth the theory of fuzzy topological spaces ($\mathcal{FTS}$) in 1968, along with some additional definitions of basic topological ideas including open set, closed set, continuity, and compactness. A thorough analysis of the construction of $\mathcal{FTS}$ was conducted by Lowen [12]. Coker [6] proposed intuitionistic fuzzy topological space ($\mathcal{IFTS}$) in 1995. Several operations on $\mathcal{IFTS}$ were described by Coker et al. [7,8].

Soft topological spaces ($\mathcal{STS}s$) was developed by Shabir and Naz [22]. Tanay and Kandemir [27] defined $\mathcal{F}_{\mathcal{S}}$ interior, $\mathcal{F}_{\mathcal{S}}$ basis, $\mathcal{F}_{\mathcal{S}}$ neighbourhood, and $\mathcal{F}_{\mathcal{S}}$ subspace topology and developed fuzzy soft topology. The theory of $\mathcal{IF}_{\mathcal{S}}$ topological space was documented in [2, 19]. A topological structure on neutrosophic soft sets was built by Bera and Mahapatra [3,4] and they examined its structural characterizations as well as the theory related to topological space, including neighbourhood, boundary interior, closure, base, subspace, continuous mappings, separation axioms, compactness and connectedness.

Zahariev [30] constructed a Matlab software package for solving fuzzy linear systems of equations and inequalities in fuzzy algebras. In 2014, Salama et al., [21] provided an Excel package for calculating neutrosophic data. Salama et al., [20] devised and implemented a neutrosophic data operation by utilizing c # programming language, Microsoft Visual Studio and NET Framework in 2014. In [11], Karunambigai and Kalaivani constructed a Matlab program

for computing the power of an intuitionistic fuzzy matrix, the index matrix and the strength of connectedness of an intuitionistic fuzzy graph. Topal et al., [28] developed hard-coded Python programs for determining the score, accuracy, certainity matrix of a bipolar neutrosophic matrix ($\mathcal{BNM}$) and for computing $\mathcal{BNM}$ intersection , $\mathcal{BNM}$ union, $\mathcal{BNM}$ complement, $\mathcal{BNM}$ addition, $\mathcal{BNM}$ product, $\mathcal{BNM}$ transpose and $\mathcal{BNM}$ composition. El-Ghareeb [10] created an Open Source Python Neutrosophic package for single-valued neutrosophic numbers and sets, interval-valued neutrosophic numbers and sets.

Sleem et al., [24] documented interval valued neutrosophic sets ($\mathcal{IVNS}$) software work that operates on $\mathcal{IVNS}$ and normalization for $\mathcal{IVNS}$ matrices. Sidiropoulos et.al., [23] presented the Python library and algorithms for fuzzy set measures. Till now researchers have developed algorithms for determining operations on neutrosophic numbers, but none of these programs can be dealt with neutrosophic soft sets or neutrosophic soft topological operators. In this context, soft code python programs have been developed for the operators in neutrosophic soft topological space in this paper.

2. Background of Neutrosophic Soft Sets

We first present the basic definitions of Neutrosophic soft sets ($\mathcal{NS}$s).

Definition 2.1. [25] Let $\mathfrak{U}$ represent the initial universe and $\mathfrak{T};\ \mathfrak{I};\ \mathfrak{F}\ :\ \mathfrak{U} \rightarrow]^{-}0, 1^{+}[$ and $^{-}0 \leq \mathfrak{T}_{\mathfrak{L}}(\xi) + \mathfrak{L}(\xi) + \mathfrak{F}_{\mathfrak{L}}(\xi) \leq\ 3^{+}$, a $\mathcal{NS}$ is written as:

$$\mathfrak{L} = \{< \xi,\ \mathfrak{T}_{\mathfrak{L}}(\xi),\ \mathfrak{I}_{\mathfrak{L}}(\xi), \mathfrak{F}_{\mathfrak{L}}(\xi) >: \xi \in \mathfrak{U}\}$$

Example 2.2. Assume that $\mathfrak{U} = \{\xi_1, \xi_2, \xi_3\}$, where ξ_1 represents the battery life, ξ_2 represents the price, ξ_3 represents the performance of an electronic scooter. The e-scooter company gives its assurance of its product. Based on the opinion of some experts regarding the company's assurance on the battery life, price and performance, the neutrosophic set can be defined as

$$(\mathfrak{N}, \Phi) = < \left\{ \frac{\xi_1}{0.7, 0.4, 0.5}, \frac{\xi_2}{0.4, 0.5, 0.5}, \frac{\xi_3}{0.3, 0.3, 0.4} \right\} >$$

Definition 2.3. [9] Let the universal set be represented by $\mathcal{U}$ and $\mathfrak{T}_{f_{\mathcal{P}(\varsigma)}}(\xi),\ \mathfrak{I}_{f_{\mathcal{P}(\varsigma)}}(\xi), \mathfrak{F}_{f_{\mathcal{P}(\varsigma)}}(\xi) \in [0, 1]$ represent the "truth", "indeterminacy", "falsity" functions of $f_{\mathcal{P}(\varsigma)}$ where $f_{\mathcal{P}} : \Phi \rightarrow \mathcal{P}(\mathcal{U})$ and Φ-set of attributes . Then the $\mathcal{NS}$ is written as $\mathcal{P} = \left\{ \left(\varsigma, \left\{ < \xi,\ \mathfrak{T}_{f_{\mathcal{P}(\varsigma)}}(\xi),\ \mathfrak{I}_{f_{\mathcal{P}(\varsigma)}}(\xi), \mathfrak{F}_{f_{\mathcal{P}(\varsigma)}}(\xi) >: \xi \in \mathcal{U} \right\} \right) : \varsigma \in \Phi \right\}$

Example 2.4. Let $\mathfrak{U}$ be the set of laptops of different companies under consideration and Φ be the set of parameters. Φ = {battery life, compact, less weight, build quality, expensive, maximum storage, RAM size, good display quality}. suppose that, there are three laptops in the universe $\mathfrak{U}$ given by $\{\xi_1, \xi_2, \xi_3\}$ and the set of parameters $\{\varsigma_1, \varsigma_2\}$, where ς_1 represents the parameter 'battery life', ς_2 represents the parameter 'expensive'.

Then the neutrosophic soft sets is written as,

$$(\mathfrak{M}, \Phi) = \left\{ \begin{array}{c} < \varsigma_1, \left\{ \frac{\xi_1}{0.9,0.4,0.3}, \frac{\xi_2}{0.5,0.3,0.5}, \frac{\xi_3}{0.4,0.1,0.3} \right\} >, \\ < \varsigma_2, \left\{ \frac{\xi_1}{0.7,0.1,0.4}, \frac{\xi_2}{0.6,0.3,0.2} \frac{\xi_3}{0.6,0.1,0.5} \right\} > \end{array} \right\}$$

Definition 2.5. [9] Let $\mathfrak{U}$ represent the initial universe and ($\mathfrak{H}$, Φ) be a $\mathcal{NSS}$ over $\mathfrak{U}$

(1) $\mathfrak{H}$ is called an absolute $\mathcal{NSS}$ if $\mathfrak{T}_{f_{\mathcal{H}(\varsigma)}}(\xi) = 1$, $\mathfrak{I}_{f_{\mathcal{H}(\varsigma)}}(\xi) = 0$, $\mathfrak{F}_{f_{\mathcal{H}(\varsigma)}}(\xi) = 0 \; \forall \; \xi \in \mathcal{U}$ and $\varsigma \in \Phi$ (written symbolically as 1_u).

(2) $\mathfrak{H}$ is called a null $\mathcal{NSS}$ if $\mathfrak{T}_{f_{\mathcal{H}(\varsigma)}}(\xi) = 0$, $\mathfrak{I}_{f_{\mathcal{H}(\varsigma)}}(\xi) = 1$, $\mathfrak{F}_{f_{\mathcal{H}(\varsigma)}}(\xi) = 1 \; \forall \; \xi \in \mathcal{U}$ and $\varsigma \in \Phi$ (written symbolically as ϕ_u).

3. Implementing Python for Computations in Neutrosophic Soft Environment

3.1. *Union and Intersection of Neutrosophic Soft Sets*

Definition 3.1. [9] For any two $\mathcal{NSS}s$ ($\mathfrak{H}$, Φ) and ($\mathfrak{G}$, Φ) over $\mathfrak{U}$, union and intersection are given by,

$$\mathfrak{H} \cup \mathfrak{G} = \mathfrak{P} = \{ \left(\varsigma, \left\{ < \xi, \; \mathfrak{T}_{f_{\mathfrak{P}(\varsigma)}}(\xi), \; \mathfrak{I}_{f_{\mathfrak{P}(\varsigma)}}(\xi), \mathfrak{F}_{f_{\mathfrak{P}(\varsigma)}}(\xi) > : \xi \in \mathfrak{U} \right\} \right) : \varsigma \in \Phi \}$$

where

$$\mathfrak{T}_{f_{\mathfrak{P}(\varsigma)}}(\xi) = \max \left(\mathfrak{T}_{f_{\mathfrak{H}(\varsigma)}}(\xi), \mathfrak{T}_{f_{\mathfrak{G}(\varsigma)}}(\xi) \right),$$

$$\mathfrak{I}_{f_{\mathfrak{P}(\varsigma)}}(\xi) = \min \left(\mathfrak{I}_{f_{\mathfrak{H}(\varsigma)}}(\xi), \mathfrak{I}_{f_{\mathfrak{G}(\varsigma)}}(\xi) \right),$$

$$\mathfrak{F}_{f_{\mathfrak{P}(\varsigma)}}(\xi) = \min \left(\mathfrak{F}_{f_{\mathfrak{H}(\varsigma)}}(\xi), \mathfrak{F}_{f_{\mathfrak{G}(\varsigma)}}(\xi) \right),$$

$$\mathfrak{H} \cap \mathfrak{G} = \mathfrak{Q} = \{ \left(\varsigma, \left\{ < \xi, \; \mathfrak{T}_{f_{\mathfrak{Q}(\varsigma)}}(\xi), \; \mathfrak{I}_{f_{\mathfrak{Q}(\varsigma)}}(\xi), \mathfrak{F}_{f_{\mathfrak{Q}(\varsigma)}}(\xi) > : \xi \in \mathfrak{U} \right\} \right) : \varsigma \in \Phi \}$$

where

$$\mathfrak{T}_{f_{\mathfrak{Q}(\varsigma)}}(\xi) = \min \left(\mathfrak{T}_{f_{\mathfrak{H}(\varsigma)}}(\xi), \mathfrak{T}_{f_{\mathfrak{G}(\varsigma)}}(\xi) \right),$$

$$\mathfrak{I}_{f_{\mathfrak{Q}(\varsigma)}}(\xi) = \max \left(\mathfrak{I}_{f_{\mathfrak{H}(\varsigma)}}(\xi), \mathfrak{I}_{f_{\mathfrak{G}(\varsigma)}}(\xi) \right),$$

$$\mathfrak{F}_{f_{\mathfrak{Q}(\varsigma)}}(\xi) = \max \left(\mathfrak{F}_{f_{\mathfrak{H}(\varsigma)}}(\xi), \mathfrak{F}_{f_{\mathfrak{G}(\varsigma)}}(\xi) \right)$$

Example 3.2. Consider $\mathfrak{U} = \{\xi_1, \xi_2\}$, the attributes $\Phi = \{\varsigma_1, \varsigma_2\}$. Let the two $\mathcal{NSS}$s $(\mathfrak{H}, \Phi)$ and $(\mathfrak{G}, \Phi)$ be given by

$$(\mathfrak{H}, \Phi) = \left\{ \begin{array}{l} < \varsigma_1, \left\{ \frac{\xi_1}{0.4,0.5,0.2}, \frac{\xi_2}{0.6,0.2,0.1} \right\} >, \\ < \varsigma_2, \left\{ \frac{\xi_1}{0.9,0.4,0.3}, \frac{\xi_2}{0.5,0.6,0.3} \right\} > \end{array} \right\}$$

$$(\mathfrak{G}, \Phi) = \left\{ \begin{array}{l} < \varsigma_1, \left\{ \frac{\xi_1}{0.8,0.3,0.4}, \frac{\xi_2}{0.2,0.4,0.6} \right\} >, \\ < \varsigma_2, \left\{ \frac{\xi_1}{0.8,0.5,0.7}, \frac{\xi_2}{0.2,0.5,0.8} \right\} > \end{array} \right\}$$

Then

$$\mathfrak{H} \cup \mathfrak{G} = \left\{ \begin{array}{l} < \varsigma_1, \left\{ \frac{\xi_1}{0.8,0.3,0.2}, \frac{\xi_2}{0.6,0.2,0.1} \right\} >, \\ < \varsigma_2, \left\{ \frac{\xi_1}{0.9,0.4,0.3}, \frac{\xi_2}{0.5,0.5,0.3} \right\} > \end{array} \right\}$$

$$\mathfrak{H} \cap \mathfrak{G} = \left\{ \begin{array}{l} < \varsigma_1, \left\{ \frac{\xi_1}{0.4,0.5,0.4}, \frac{\xi_2}{0.2,0.4,0.6} \right\} >, \\ < \varsigma_2, \left\{ \frac{\xi_1}{0.8,0.5,0.7}, \frac{\xi_2}{0.2,0.6,0.8} \right\} > \end{array} \right\}$$

3.1.1. *Function definition*

The following functions define the union and intersection of any two $\mathcal{NSS}s$

```
def nss_union(l1, l2):
    """Return tuple comprised of the maximum, minimum,
    minimum value in each tuple argument."""
    result = [[((max(l1[g][d][0], l2[g][d][0])),
                (min(l1[g][d][1], l2[g][d][1])),
                (min(l1[g][d][2], l2[g][d][2])))
        for d in range(elements)] for g in range(attributes)]
    return (result)
```

```
def nss_intersection(l1, l2):
    """Return tuple comprised of the minimum, maximum,
    maximum value in each tuple argument."""
    result = [[((min(l1[g][d][0], l2[g][d][0])),
                (max(l1[g][d][1], l2[g][d][1])),
                (max(l1[g][d][2], l2[g][d][2])))
        for d in range(elements)] for g in range(attributes)]
    return (result)
```

3.1.2. *Program and intuitive description*

```
from pprint import pprint

attributes = int(input("Enter the number of attributes : "))
elements = int(input("Enter the number of elements in the universal
    ↪ set : "))

print("Enter the elements of N_s set :A")
l1 = [[tuple(map(float, input().split(" "))) for x in range(
    ↪ elements)] for g in range(attributes)]

print("Enter the elements of N_s set :B")
l2= [[tuple(map(float, input().split(" "))) for x in range(elements
    ↪ )] for g in range(attributes)]

def nss_union(l1, l2):
    """Return tuple comprised of the maximum, minimum, minimum
        ↪ value in each tuple argument."""
    result = [[((max(l1[g][d][0], l2[g][d][0])),
                (min(l1[g][d][1], l2[g][d][1])),
                (min(l1[g][d][2], l2[g][d][2]))) for d in range(
                    ↪ elements)] for g in range(attributes)]
    return (result)

print("Union of Neutrosophic soft sets is")
pprint(nss_union(l1, l2))

def nss_intersection(l1, l2):
    """Return tuple comprised of the minimum, maximum, maximum
        ↪ value in each tuple argument."""
    result = [[((min(l1[g][d][0], l2[g][d][0])),
                (max(l1[g][d][1], l2[g][d][1])),
                (max(l1[g][d][2], l2[g][d][2]))) for d in range(
                    ↪ elements)] for g in range(attributes)]
    return (result)
```

```
print("Intersection of Neutrosophic soft sets is")
pprint(nss_intersection(l1, l2))
```

The program initially gets the value of no. of. attributes and no. of. elements in the universal set and stores in the variable namely attributes and elements respectively. Gets the elements of neutrosophic soft sets. Compute the union and intersection of the $\mathcal{NSS}$s using the defined functions.

```
Enter the number of attributes : 2
Enter the number of elements in the universal set: 2
Enter the elements of N_s set:A
.4 .5 .2
.6 .2 .1
.9 .4 .3
.5 .6 .3
Enter the elements of N_s set:B
.8 .3 .4
.2 .4 .6
.8 .5 .7
.2 .5 .8
Union of Neutrosophic soft sets is
[[(0.8, 0.3, 0.2), (0.6, 0.2, 0.1)], [(0.9, 0.4, 0.3), (0.5, 0.5, 0.3)]]
Intersection of Neutrosophic soft sets is
[[(0.4, 0.5, 0.4), (0.2, 0.4, 0.6)], [(0.8, 0.5, 0.7), (0.2, 0.6, 0.8)]]
```

FIGURE 1. Output of the union and intersection of neutrosophic soft sets program

3.2. *Neutrosophic Soft Subset and Superset*

Definition 3.3. [9] For any two $\mathcal{NSS}s$ $(\mathfrak{H}, \Phi)$ and $(\mathfrak{G}, \Phi)$ over $\mathfrak{U}$,

(1) $(\mathfrak{H}, \Phi) \subseteq (\mathfrak{G}, \Phi)$ if $\mathfrak{T}_{\mathfrak{H}(\varsigma)}(\xi) \leq \mathfrak{T}_{\mathfrak{G}(\varsigma)}(\xi)$, $\mathfrak{I}_{\mathfrak{H}(\varsigma)}(\xi) \geq \mathfrak{I}_{\mathfrak{G}(\varsigma)}(\xi)$, $\mathfrak{F}_{\mathfrak{H}(\varsigma)}(\xi) \geq \mathfrak{F}_{\mathfrak{G}(\varsigma)}(\xi)$, $\forall\, \varsigma \in \Phi,\ \xi \in \mathfrak{U}$.

(2) $(\mathfrak{H}, \Phi) \supseteq (\mathfrak{G}, \Phi)$ if $\mathfrak{T}_{\mathfrak{H}(\varsigma)}(\xi) \geq \mathfrak{T}_{\mathfrak{G}(\varsigma)}(\xi)$, $\mathfrak{I}_{\mathfrak{H}(\varsigma)}(\xi) \leq \mathfrak{I}_{\mathfrak{G}(\varsigma)}(\xi)$, $\mathfrak{F}_{\mathfrak{H}(\varsigma)}(\xi) \leq \mathfrak{F}_{\mathfrak{G}(\varsigma)}(\xi)$, $\forall\, \varsigma \in \Phi,\ \xi \in \mathfrak{U}$.

3.2.1. *Function definition*

The following defined functions identify whether the given $\mathcal{NSS}$ is a neutrosophic soft subset or superset of another $\mathcal{NSS}$.

```
def nss_sub(l1, l2):
    for g in range(attributes):
        for d in range(elements):
            if (((l1[g][d][0] <= l2[g][d][0]) and
            (l1[g][d][1] >= l2[g][d][1]) and
            (l1[g][d][2] >= l2[g][d][2]))):
                return 1
            return 0
```

The function returns the value 1, if the condition is true, else it returns the value 0.

```
def nss_sup(l1, l2):
    for g in range(attributes):
        for d in range(elements):
            if (((l1[g][d][0] >= l2[g][d][0]) and
            (l1[g][d][1] <= l2[g][d][1]) and
            (l1[g][d][2] <= l2[g][d][2]))):
                return 1
            return 0
```

The function returns the value 1, if the condition is true, else it returns the value 0.

3.3. *Complement of a Neutrosophic Soft Set*

Definition 3.4. [9] For any $\mathcal{NSS}$ $(\ \mathfrak{H},\ \Phi\ )$ over $\mathfrak{U}$, the complement is given by,

$$\mathfrak{H}^c = \left\{\left(\varsigma, \left\{< \xi,\ \mathfrak{F}_{f_{\mathfrak{H}(\varsigma)}}(\xi), 1 -\ \mathfrak{I}_{f_{\mathfrak{H}(\varsigma)}}(\xi), \mathfrak{T}_{f_{\mathfrak{H}(\varsigma)}}(\xi) >: \xi \in \mathfrak{U}\right\}\right) : \varsigma \in \Phi\right\}$$

3.3.1. *Function definition*

The following defined function gives the complement of a $\mathcal{NSS}$

```
def nss_compl(m):
    """Return tuple comprised of the complement value
    in each tuple argument."""
    result = [[((m[g][d][2]),
            (round((1 - m[g][d][1]), 1)),
            (m[g][d][0])) for d in range(elements)]
            for g in range(attributes)]
    return (result)
```

3.3.2. *Program and intuitive description*

```
from pprint import pprint

attributes = int(input("Enter the number of attributes : "))
elements = int(input("Enter the number of elements in the universal
    ↪ set : "))

print("Enter the elements of N_s set :")
A = [[tuple(map(float, input().split(" "))) for d in range(elements
    ↪ )] for g in range(attributes)]

def nss_compl(m):
    """Return tuple comprised of the complement value in each tuple
       ↪ argument."""
    result = [[((m[g][d][2]), (round((1 - m[g][d][1]), 1)), (m[g][d
       ↪ ][0])) for d in range(elements)] for g in range(attributes
       ↪ )]
    return (result)

print("The complement of the Neutrosophic soft set is :")
pprint(nss_compl(A))
```

The program initially gets the value of the number of attributes and number of elements in the universal set and stores it in the variable namely attributes and elements respectively. Gets the elements of $\mathcal{NSS}$s. Compute the complement of the $\mathcal{NSS}$s using the defined functions.

```
Enter the number of attributes : 2
Enter the number of elements in the universal set: 2
Enter the elements of N_s set:
.4 .5 .2
.6 .2 .1
.9 .4 .3
.5 .6 .3
The complement of the Neutrosophic soft set is:
[[(0.2, 0.5, 0.4), (0.1, 0.8, 0.6)], [(0.3, 0.6, 0.9), (0.3, 0.4, 0.5)]]
```

FIGURE 2. Output of the complement of a neutrosophic soft set program

3.4. *Neutrosophic Soft Topology*

Definition 3.5. [3] Let the family of all $\mathcal{NS}$s over $\mathfrak{U}$ via parameters in Φ be represented as $\mathcal{NS}(\mathfrak{U}, \Phi)$ and $\tau_u \subset \mathcal{NS}(\mathfrak{U}, \Phi)$. Then τ_u is known as a $\mathcal{NS}$ topology on $(\mathfrak{U},\ \Phi)$ if the following conditions are true.

(1) $\phi_u,\ 1_u \in \tau_u$
(2) $\forall\ \mathfrak{D}_1, \mathfrak{D}_2 \in \tau_u \Rightarrow \mathfrak{D}_1 \cap \mathfrak{D}_2 \in \tau_u$
(3) for $\cup_{i\in J}\mathfrak{D}_i \in \tau_u$, $\forall\ \{\mathfrak{D}_i : i \in J\} \subseteq \tau_u$

Then $(\mathfrak{U},\ \Phi,\ \tau_u)$ is called as a neutrosophic soft topological space ($\mathcal{NTS}$). Every member of τ_u is called as neutrosophic soft open set ($\mathcal{NOS}$) and its complement is termed as neutrosophic soft closed set ($\mathcal{NCS}$).

Example 3.6. Let $\mathfrak{U} = \{\xi_1, \xi_2, \xi_3\}$ and the attributes $\Phi = \{\varsigma_1, \varsigma_2\}$. Consider the $\mathcal{NS}$s

$$(\mathfrak{N}, \Phi) = \left\{ \begin{array}{l} < \varsigma_1, \left\{ \frac{\xi_1}{0.7,0.4,0.5}, \frac{\xi_2}{0.4,0.5,0.5}, \frac{\xi_3}{0.3,0.3,0.4} \right\} >, \\ < \varsigma_2, \left\{ \frac{\xi_1}{0.6,0.2,0.4}, \frac{\xi_2}{0.5,0.4,0.3}, \frac{\xi_3}{0.4,0.6,0.5} \right\} > \end{array} \right\}$$

$$(\mathfrak{P}, \Phi) = \left\{ \begin{array}{l} < \varsigma_1, \left\{ \frac{\xi_1}{0.8,0.3,0.4}, \frac{\xi_2}{0,5,0.4,0.3}, \frac{\xi_3}{0.7,0.1,0.2} \right\} >, \\ < \varsigma_2, \left\{ \frac{\xi_1}{0.7,0.1,0.3}, \frac{\xi_2}{0.6,0.2,0.1}, \frac{\xi_3}{0.7,0.4,0.3} \right\} > \end{array} \right\}$$

Then, $\tau_u = \{\ \phi_u,\ 1_u,\ (\mathfrak{N}, \Phi), (\mathfrak{P}, \Phi)\}$ forms a $\mathcal{NS}$ topology.

3.4.1. *Program and intuitive description*

```
from pprint import pprint
import numpy as np

print("Enter the collection of Neutrosophic Soft Sets along with
    ↪ null and absolute Neutrosophic Soft Sets")
m = int(input("Number of Neutrosophic Soft Sets: "))
attributes = int(input("Enter the Number of attributes : "))
```

```
elements = int(input("Enter the Number of elements in the Universe
    ↪ Set: "))

PNS_Sets = []
a = [[0] * elements for i in range(attributes)]
for i in range(m):
    print("Enter the Primary Neutrosophic soft set:", i + 1)
    ns_set = [[tuple(map(float, input().split(" "))) for d in range
        ↪ (elements)]
                for g in range(attributes)]
    PNS_Sets.append(ns_set)

def nss_compl(m):
    """Return tuple comprised of the complement values in each tuple
        ↪ argument."""
    result = [[((m[g][d][2]), (round((1 - m[g][d][1]), 1)), (m[g][d
        ↪ ][0])) for d in range(elements)] for g in range(attributes
        ↪ )]
    return (result)

def ns_union(m1, m2):
    """Return tuple comprised of the max, min, min value in each
        ↪ tuple argument."""
    result = [[((max(m1[g][d][0], m2[g][d][0])), (min(m1[g][d][1],
        ↪ m2[g][d][1])),
                (min(m1[g][d][2], m2[g][d][2]))) for d in range(
                    ↪ elements)] for g in range(attributes)]
    return (result)

def ns_int(m1, m2):
    """Return tuple comprised of the min, max, max value in each
        ↪ tuple argument."""
    result = [[((min(m1[g][d][0], m2[g][d][0])), (max(m1[g][d][1],
        ↪ m2[g][d][1])),
                (max(m1[g][d][2], m2[g][d][2]))) for d in range(
                    ↪ elements)] for g in range(attributes)]
```

```
    return (result)

Union = []
for m1 in range(len(PNS_Sets)):
    for m2 in range(len(PNS_Sets)):
        x1 = ns_union(PNS_Sets[m1], PNS_Sets[m2])
        Union.append(x1)

res = [Union[i] for i in range(len(Union)) if i == Union.index(
    ↪ Union[i]) ]

Intersection = []
for m1 in range(len(PNS_Sets)):
    for m2 in range(len(PNS_Sets)):
        x2 = ns_int(PNS_Sets[m1], PNS_Sets[m2])
        Intersection.append(x2)

res1 = [Intersection[i] for i in range(len(Intersection)) if i ==
    ↪ Intersection.index(Intersection[i]) ]

Topology = []
for m1 in res:
    Topology.append(m1)
for m2 in res1:
    Topology.append(m2)

topology = []
for m1 in range(len(Topology)):
    for m2 in range(len(Topology)):
        y1 = ns_union(Topology[m1], Topology[m2])
        y2 = ns_int(Topology[m1], Topology[m2])
        topology.append(y1)
        topology.append(y2)

print("The Neutrosophic Soft Topology is:")
```

```
res3 = [topology[i] for i in range(len(topology)) if i == topology.
    ↪ index(topology[i]) ]
pprint(res3)
```

The above program is developed for framing $\mathcal{NS}$ topology using the numpy package. Program initially gets the values of the number of $\mathcal{NS}$s, number of attributes and number of elements. As the second step, $\mathcal{NS}$s are collected as input and stored in the list called PNS_Sets. The $\mathcal{NS}$-complement, $\mathcal{NS}$-union function and $\mathcal{NS}$-intersection function are defined for computing complement, union and intersection of $\mathcal{NS}$s respectively. In the third step, the program creates a list of union and intersection of the $\mathcal{NS}$s in the list PNS_Sets and are appended to the list "topology". The result is finally printed using pprint module.

```
Enter the collection of Neutrosophic Soft Sets along with null and absolute Neutrosophic Soft Sets
Number of Neutrosophic Soft Sets: 6
Enter the Number of attributes : 2
Enter the Number of elements in the Universe Set: 2
Enter the Primary Neutrosophic soft set: 1
1 0 0
1 0 0
1 0 0
1 0 0
Enter the Primary Neutrosophic soft set: 2
0 1 1
0 1 1
0 1 1
0 1 1
Enter the Primary Neutrosophic soft set: 3
.8 .3 .4
.5 .4 .3
.7 .1 .2
.7 .1 .3
Enter the Primary Neutrosophic soft set: 4
.3 .3 .4
.6 .2 .4
.5 .4 .3
.4 .6 .5
Enter the Primary Neutrosophic soft set: 5
.1 .9 .8
.2 .9 .8
.1 .8 .7
.3 .7 .8
Enter the Primary Neutrosophic soft set: 6
.3 .3 .4
.6 .2 .4
.5 .4 .3
.4 .6 .5
The Neutrosophic Soft Topology is:
[[[(1.0, 0.0, 0.0), (1.0, 0.0, 0.0)], [(1.0, 0.0, 0.0), (1.0, 0.0, 0.0)]],
 [[(0.0, 1.0, 1.0), (0.0, 1.0, 1.0)], [(0.0, 1.0, 1.0), (0.0, 1.0, 1.0)]],
 [[(0.8, 0.3, 0.4), (0.5, 0.4, 0.3)], [(0.7, 0.1, 0.2), (0.7, 0.1, 0.3)]],
 [[(0.3, 0.3, 0.4), (0.6, 0.2, 0.4)], [(0.5, 0.4, 0.3), (0.4, 0.6, 0.5)]],
 [[(0.1, 0.9, 0.8), (0.2, 0.9, 0.8)], [(0.1, 0.8, 0.7), (0.3, 0.7, 0.8)]],
 [[(0.8, 0.3, 0.4), (0.6, 0.2, 0.3)], [(0.7, 0.1, 0.2), (0.7, 0.1, 0.3)]],
 [[(0.3, 0.3, 0.4), (0.5, 0.4, 0.4)], [(0.5, 0.4, 0.3), (0.4, 0.6, 0.5)]]]
```

FIGURE 3. Output of the Neutrosophic Soft Topology framing program

3.4.2. *Program and intuitive description*

```
from pprint import pprint
import numpy as np

print("Enter the collection of Neutrosophic Soft Sets")
m = int(input("Number of Neutrosophic Soft Sets: "))
attributes = int(input("Enter the Number of attributes : "))
elements = int(input("Enter the Number of elements in the Universe
    ↪ Set: "))

NS_Sets = []
a = [[0] * elements for i in range(attributes)]
for i in range(m):
    print("Enter the Neutrosophic soft set:", i + 1)
    ns_set = [[tuple(map(float, input().split(" "))) for d in range
        ↪ (elements)] for g in range(attributes)]
    NS_Sets.append(ns_set)

def ns_union(n1, n2):
    """Return tuple comprised of the max, min, min value in each
        ↪ tuple argument."""
    result = [[((max(n1[g][d][0], n2[g][d][0])),
                (min(n1[g][d][1], n2[g][d][1])),
                (min(n1[g][d][2], n2[g][d][2]))) for d in range(
                    ↪ elements)] for g in range(attributes)]
    return (result)

def ns_int(n1, n2):
    """Return tuple comprised of the min, max, max value in each
        ↪ tuple argument."""
    result = [[((min(n1[g][d][0], n2[g][d][0])),
                (max(n1[g][d][1], n2[g][d][1])),
                (max(n1[g][d][2], n2[g][d][2]))) for d in range(
                    ↪ elements)] for g in range(attributes)]
    return (result)
```

```
Topology = []
for n1 in NS_Sets:
    for n2 in NS_Sets:
        if (np.array_equal(n1, n2) == False):
            x1 = ns_union(n1, n2)
            x2 = ns_int(n1, n2)
            res1 = any(np.array_equal(x1, m) for m in NS_Sets)
            res2 = any(np.array_equal(x2, m) for m in NS_Sets)
            Topology.append(res1)
            Topology.append(res2)

if all(Topology) == True:
    print("The Collection forms a Neutrosophic Soft Topology")
else:
    print("Does not form a Neutrosophic Soft Topology")
```

The above program is developed for verifying whether a forms a $\mathcal{N}_{\mathcal{S}}$ topology or not. The program initially gets the values of the number of $\mathcal{N}_{\mathcal{S}}\mathcal{OS}$s, number of attributes and number of elements. In the second step, $\mathcal{N}_{\mathcal{S}}\mathcal{OS}$s are collected as input and stored in the list called NS_Sets . The $\mathcal{N}_{\mathcal{S}}$-union function and $\mathcal{N}_{\mathcal{S}}$-intersection function are defined for computing union and intersection of $\mathcal{N}_{\mathcal{S}}\mathcal{OS}$s respectively.

In the third step, the program creates a list of Boolean values. The list is created by taking two $\mathcal{N}_{\mathcal{S}}\mathcal{OS}$s from the list of $\mathcal{N}_{\mathcal{S}}\mathcal{OS}$s and checking the condition if the two $\mathcal{N}_{\mathcal{S}}\mathcal{OS}$s are equal, if they are equal then exists the loop, if not then the program computes the values of $\mathcal{N}_{\mathcal{S}}$ union and $\mathcal{N}_{\mathcal{S}}$ intersection of the two $\mathcal{N}_{\mathcal{S}}\mathcal{OS}$s using pre-defined $\mathcal{N}_{\mathcal{S}}$-union and $\mathcal{N}_{\mathcal{S}}$-intersection function. In the next step, the program checks whether the computed values of $\mathcal{N}_{\mathcal{S}}$ intersection and $\mathcal{N}_{\mathcal{S}}$ union are in the list of $\mathcal{N}_{\mathcal{S}}\mathcal{OS}$s utilizing the built-in 'any' function which returns only Boolean values as result. The results are appended to the list 'Topology'

In the final step, the program checks if all the values in the list 'Topology' are true utilizing the built-in 'all' function, if the result is true then the collection of input values forms a $\mathcal{N}_{\mathcal{S}}\mathcal{TS}$ or else it does not.

```
Enter the collection of Neutrosophic Soft Sets
Number of Neutrosophic Soft Sets: 7
Enter the Number of attributes : 2
Enter the Number of elements in the Universe Set: 2
Enter the Neutrosophic soft set: 1
1 0 0
1 0 0
1 0 0
1 0 0
Enter the Neutrosophic soft set: 2
0 1 1
0 1 1
0 1 1
0 1 1
Enter the Neutrosophic soft set: 3
.8 .3 .4
.5 .4 .3
.7 .1 .2
.7 .1 .3
Enter the Neutrosophic soft set: 4
.3 .3 .4
.6 .2 .4
.5 .4 .3
.4 .6 .5
Enter the Neutrosophic soft set: 5
.1 .9 .8
.2 .9 .8
.1 .8 .7
.3 .7 .8
Enter the Neutrosophic soft set: 6
.8 .3 .4
.6 .2 .3
.7 .1 .2
.7 .1 .3
Enter the Neutrosophic soft set: 7
.3 .3 .4
.5 .4 .4
.5 .4 .3
.4 .6 .5
The Collection forms a Neutrosophic Soft Topology
```

FIGURE 4. Output of the Neutrosophic Soft Topology program

3.5. *Operators of Neutrosophic Soft Topological Space*

Definition 3.7. [3] Let $\mathcal{NSTS}$ be represented as $(\mathfrak{U}, \Phi, \tau_u)$ and $(\mathfrak{L}, \Phi)$ be any arbitrary $\mathcal{NSS}$. Then the closure of a $\mathcal{NSS}$ $(\mathfrak{L}, \Phi)$ is the intersection of all $\mathcal{NSCSs}$ containing $(\mathfrak{L}, \Phi)$.

i.e., $\mathcal{NS}cl(\mathfrak{L}, \Phi) = \cap\{(\mathfrak{F}, \Phi) : (\mathfrak{L}, \Phi) \subseteq (\mathfrak{F}, \Phi)$, $(\mathfrak{F}, \Phi)$ is a $\mathcal{NSCS}$ in $\mathfrak{U}.\}$

Definition 3.8. [3] Let $\mathcal{NSTS}$ be represented as $(\mathfrak{U}, \Phi, \tau_u)$ and $(\mathfrak{L}, \Phi)$ be any arbitrary $\mathcal{NSS}$. Then the interior of a $\mathcal{NSS}$ $(\mathfrak{L}, \Phi)$ is the union of all $\mathcal{NSOSs}$ contained in $(\mathfrak{L}, \Phi)$.

i.e., $\mathcal{NS}int(\mathfrak{L}, \Phi) = \cup\{(\mathfrak{G}, \Phi) : (\mathfrak{G}, \Phi) \subseteq (\mathfrak{L}, \Phi)$, $(\mathfrak{G}, \Phi)$ is a $\mathcal{NSOS}$ in $\mathfrak{U}$ $\}$.

Example 3.9. Consider the Example 3.6,

$$(\mathfrak{N},\Phi)^c = \left\{ \begin{array}{l} < \varsigma_1, \left\{ \frac{\xi_1}{0.5,0.6,0.7}, \frac{\xi_2}{0.5,0.5,0.4}, \frac{\xi_3}{0.4,0.7,0.3} \right\} >, \\ < \varsigma_2, \left\{ \frac{\xi_1}{0.4,0.8,0.6}, \frac{\xi_2}{0.3,0.6,0.5}, \frac{\xi_3}{0.5,0.4,0.4} \right\} > \end{array} \right\}$$

$$(\mathfrak{P},\Phi)^c = \left\{ \begin{array}{l} < \varsigma_1, \left\{ \frac{\xi_1}{0.4,0.7,0.8}, \frac{\xi_2}{0,3,0.6,0.5}, \frac{\xi_3}{0.2,0.9,0.7} \right\} >, \\ < \varsigma_2, \left\{ \frac{\xi_1}{0.3,0.9,0.7}, \frac{\xi_2}{0.1,0.8,0.6}, \frac{\xi_3}{0.3,0.6,0.7} \right\} > \end{array} \right\}$$

Then, $\tau_u{}^c = \{ \phi_u,\ 1_u,\ (\mathfrak{N},\Phi)^c, (\mathfrak{P},\Phi)^c \}$ is the collection of $\mathcal{NSCS}$s.

Let $(\mathfrak{S},\Phi)$ be an arbitrary $\mathcal{NSS}$ defined as

$$(\mathfrak{S},\Phi) = \left\{ \begin{array}{l} < \varsigma_1, \left\{ \frac{\xi_1}{0.2,0.8,0.9}, \frac{\xi_2}{0.3,0.7,0.7}, \frac{\xi_3}{0.1,0.9,0.8} \right\} >, \\ < \varsigma_2, \left\{ \frac{\xi_1}{0.2,0.9,0.8}, \frac{\xi_2}{0.1,0.8,0.7}, \frac{\xi_3}{0.3,0.7,0.8} \right\} > \end{array} \right\}$$

Here $\mathcal{NS}cl(\mathfrak{S},\Phi) = (\mathfrak{P},\Phi)^c$ and $\mathcal{NS}int(\mathfrak{S},\Phi) = \phi_u$

3.5.1. *Program and intuitive description*

```
from pprint import pprint

m = int(input("Number of Neutrosophic Soft Open Sets: "))
attributes = int(input("Enter the Number of attributes : "))
elements = int(input("Enter the Number of elements in the Universe
    ↪ Set: "))

NS_Sets = []
p = [[0] * elements for g in range(attributes)]
for i in range(m):
    print("Enter the Neutrosophic soft set:", i + 1)
    ns_set = [[tuple(map(float, input().split(" "))) for e in range
        ↪ (elements)]
                for a in range(attributes)]
    NS_Sets.append(ns_set)

def nss_compl(m):
    """Return tuple comprised of the complement value in each tuple
        ↪ argument."""
    result = [[((m[a][e][2]), (round((1 - m[a][e][1]), 1)), (m[a][e
        ↪ ][0])) for e in range(elements)] for a in range(attributes
        ↪ )]
    return (result)
```

```
complement = []
for m in NS_Sets:
    x = nss_compl(m)
    complement.append(x)

mA = int(input("Number of Arbitrarty Neutrosophic Soft Sets: "))

Arbitrary_Sets = []
p = [[0] * elements for i in range(attributes)]
for i in range(mA):
    print("Enter the Artbitrary Neutrosophic soft set:", i + 1)
    arbi_set = [[tuple(map(float, input().split(" "))) for e in
        ↪ range(elements)]
                for a in range(attributes)]
    Arbitrary_Sets.append(arbi_set)

def superset(A, m):
    return (((m[g][d][0] >= A[g][d][0]) and (m[g][d][1] <= A[g][d
        ↪ ][1]) and (m[g][d][2] <= A[g][d][2])))

def subset(A, m):
    return (((m[g][d][0] <= A[g][d][0]) and (m[g][d][1] >= A[g][d
        ↪ ][1]) and (m[g][d][2] >= A[g][d][2])))

def nss_union(m1, m2):
    """Return tuple comprised of the max, min, min value in each
        ↪ tuple argument."""
    result = [[((max(m1[g][d][0], m2[g][d][0])), (min(m1[g][d][1],
        ↪ m2[g][d][1])),
                (min(m1[g][d][2], m2[g][d][2]))) for d in range(
                    ↪ elements)] for g in range(attributes)]
    return (result)

def nss_intersection(m1, m2):
```

```
    """Return tuple comprised of the min, max, max value in each
       ↪ tuple argument."""
    result = [[((min(m1[g][d][0], m2[g][d][0])), (max(m1[g][d][1],
       ↪ m2[g][d][1])),
               (max(m1[g][d][2], m2[g][d][2]))) for d in range(
                  ↪ elements)] for g in range(attributes)]
    return (result)

for m1 in Arbitrary_Sets:
    Superset = []
    for m in complement:
        cal1 = []
        for g in range(attributes):
            for d in range(elements):
                v = superset(m1, m)
                cal1.append(v)
        if all(cal1):
            Superset.append(m)
    closure = Superset[0]
    for i in range(len(Superset)):
        closure = nss_intersection(closure, Superset[i])
    print("Closure:")
    pprint(closure)

for m1 in Arbitrary_Sets:
    Subset = []
    for m in NS_Sets:
        cal2 = []
        for g in range(attributes):
            for d in range(elements):
                l = subset(m1, m)
                cal2.append(l)
        if all(cal2):
            Subset.append(m)
    interior = Subset[0]
    for i in range(len(Subset)):
```

```
        closure = nss_union(interior, Subset[i])
    print("Interior:")
    pprint(interior)
```

The $\mathcal{NS}$-complement, $\mathcal{NS}$-superset, $\mathcal{NS}$-subset $\mathcal{NS}$-union and $\mathcal{NS}$-intersection functions are defined in this program. Program initially gets the values of the number of $\mathcal{NSOS}$s, number of attributes and number of elements. In the second step, $\mathcal{NSOS}$s are collected as input and stored in the list called NS_Sets. Then it creates the list of complement using the $\mathcal{NS}$-Complement function. In the third step, the program gets the arbitrary $\mathcal{NSS}s$ and creates a list of arbitrary $\mathcal{NSS}s$.

For each element in the list of arbitrary $\mathcal{NSS}$, the program compiles a list of supersets (subsets) by taking $\mathcal{NSCS}$s ($\mathcal{NSOS}$s) from the list of complement ($\mathcal{NSOS}s$) using the pre-defined superset (subset) function. Then the program computes the closure (interior) of the arbitrary set by taking $\mathcal{NSS}$ from the list of supersets (subsets) and using the $\mathcal{NS}$-intersection ($\mathcal{NS}$-union) function, prints the value of the closure(interior) of the arbitrary set.

```
Number of Neutrosophic Soft Open Sets: 4
Enter the Number of attributes : 2
Enter the Number of elements in the Universe Set: 3
Enter the Neutrosophic soft set: 1
.7 .4 .5
.4 .5 .5
.3 .3 .4
.6 .2 .4
.5 .4 .3
.4 .6 .5
Enter the Neutrosophic soft set: 2
.8 .3 .4
.5 .4 .3
.7 .1 .2
.7 .1 .3
.6 .2 .1
.7 .4 .3
Enter the Neutrosophic soft set: 3
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
Enter the Neutrosophic soft set: 4
0 1 1
0 1 1
0 1 1
0 1 1
0 1 1
0 1 1
Number of Arbitrarty Neutrosophic Soft Sets: 1
Enter the Artbitrary Neutrosophic soft set: 1
.2 .8 .9
.3 .7 .7
.1 .9 .8
.2 .9 .8
.1 .8 .7
.3 .7 .8
```

```
Number of Arbitrarty Neutrosophic Soft Sets: 1
Enter the Artbitrary Neutrosophic soft set: 1
.2 .8 .9
.3 .7 .7
.1 .9 .8
.2 .9 .8
.1 .8 .7
.3 .7 .8
Closure:
[[(0.4, 0.7, 0.8), (0.3, 0.6, 0.5), (0.2, 0.9, 0.7)],
 [(0.3, 0.9, 0.7), (0.1, 0.8, 0.6), (0.3, 0.6, 0.7)]]
Interior:
[[(0.0, 1.0, 1.0), (0.0, 1.0, 1.0), (0.0, 1.0, 1.0)],
 [(0.0, 1.0, 1.0), (0.0, 1.0, 1.0), (0.0, 1.0, 1.0)]]
```

FIGURE 5. Output of closure and interior of a neutrosophic soft set program

4. **Conclusion**

In this paper, python functions for union, intersection, superset, subset and complement for working in $\mathcal{N}_{\mathcal{S}}$ environment were defned and python program for union, intersection and complement in $\mathcal{N}_{\mathcal{S}}$ environment were furnished. Further python program for computing closure and interior of a $\mathcal{N}_{\mathcal{S}}\mathcal{S}$ in $\mathcal{N}_{\mathcal{S}}$ environment was provided. Moreover, Python program was constructed to verify whether a particular collection of $\mathcal{N}_{\mathcal{S}}\mathcal{S}$s forms a $\mathcal{N}_{\mathcal{S}}$-topology or not. User HTML interface can be developed using Pyscript in near future. In due course, the python programme can be built to find the optimal solution to any decision-making problem with complex data.

References

1. Atanassov, K. Intuitionistic fuzzy sets, Fuzzy Sets and Systems, (1986), 20(1), pp. 87 — 96 . DOI: 10.1007/978-3-7908-1870-3-1.
2. Bayramov, S. and Gunduz, C. On intuitionistic fuzzy soft topological spaces, TWMS J Pure Appl Math, (2014), 5(1), pp.66-79.
3. Bera, T; N. K. Mahapatra, N. K. Introduction to Neutrosophic Soft Topological Space, Opsearch, (2017), 54(4), pp. 841 — 867. DOI: 10.1007/s12597-017-0308-7.
4. Bera, T; N. K. Mahapatra, N. K. On neutrosophic soft topological space. Neutrosophic sets and systems, (2018), 19(1), pp. 3–15.
5. Chang, C. L. Fuzzy topological spaces, Journal of Mathematical Analysis and Applications, (1968), 24(1), pp. 82—190.
6. Çoker, D. An introduction to intuitionistic fuzzy topological spaces. *Fuzzy Sets and Systems*, **1997**, *88(1)*, 81—89.
7. Coker, D; Haydar Es, A. On fuzzy compactness in intuitionistic fuzzy topological spaces, Journal of Fuzzy Mathematics, (1995), 3, pp. 899—910.
8. Coker, D; Turanli, N. Fuzzy connectedness in intuitionistic fuzzy topological spaces, Fuzzy Sets and Systems, (2000), 116(3), pp. 369—375.
9. Deli, I; Broumi, S. Neutrosophic soft matrices and NSM-decision making, Journal of Intelligent and Fuzzy Systems, (2015), 28(5), pp. 2233-–2241.

10. El-Ghareeb, H. A. Novel Open Source Python Neutrosophic Package. Neutrosophic Sets and Systems, (2019), 25, pp. 136–160. DOI: 10.5281/zenodo.2631514.
11. Karunambigai, M. G.; Kalaivani, O. K. Software development in intuitionistic Fuzzy Relational Calculus. International Journal of Scientific and research Publication, (2016), 6(7), pp. 311–331.
12. Lowen, R. Fuzzy topological spaces and fuzzy compactness, Journal of Mathematical Analysis and Applications, (1976), 56(3), pp. 621-–633.
13. Maji, P. K.; Biswas, R.; Roy, A. R. Fuzzy soft sets, Journal of Fuzzy Mathematics,(2001), 9(3), pp. 589—602 .
14. Maji, P. K.; Biswas, R.; Roy, A. R. Intuitionistic fuzzy soft sets, Journal of Fuzzy Mathematics, (2001), 9(3), pp. 677–692.
15. Maji, P. K.; Biswas, R.; Roy, A. R. An application of soft sets in a decision making problem, Computers & Mathematics With Applications, (2002), 44, pp. 1077—1083.
16. Maji, P. K.;Biswas, R; Roy, R. Soft set theory, Computers & Mathematics with Applications., (2003), 45, pp. 555—562.
17. Maji, P. K. Neutrosophic soft set, Annals of Fuzzy Mathematics and Informatics, (2013), 5(1), pp. 157—168.
18. Molodtsov, D. Soft set theory: first results. Computers & Mathematics with Applications, (1999), 37, pp. 19–31.
19. Osmanoglu, I; Tokat, D. On intuitionistic Fuzzy soft topology, General Mathematics Notes, (2013), 19(2), pp. 59—70.
20. Salama, A. A.;Abd El Fattah, A.; El-Ghareeb, H.; Manie, A. M. Design and implementation of neutrosophic data operations using object oriented programming. International Journal of Computer Application, (2014), 5 (4),pp. 163—175.
21. Salama, A. A.; El-Ghareeb, H. A.; Manie, A. M.; Smarandache, F. Introduction to Develop Some Software Programs for Dealing with Neutrosophic Sets, Neutrosophic Sets and Systems, (2014), 4, pp. 53–54.
22. Shabir, M.; Naz, M. On soft topological spaces, Computers & Mathematics with Applications, (2011), 61(8),pp. 1786—1799.
23. Sidiropoulos, G. K; Apostolidis, K. D., Damianos, N.; Papakostas, G. A. Fsmpy: A Fuzzy Set Measures Python Library, Information, (2022), 13(2), 64.
24. Sleem, A; Mohamed Abdel-Baset; Ibrahim El-henawy. PyIVNS: A python based tool for Interval-valued neutrosophic operations and normalization, SoftwareX, (2020) 12.
25. Smarandache, F. Neutrosophic set, A Generalisation of the Intuitionistic Fuzzy Sets, International Journal of Pure and Applied Mathematics, (2005), 24(3), pp. 287—297.
26. Smarandache, F. Neutrosophy, Neutrosophic Probability, Set and Logic. American Research Press, Rehoboth, USA, (1998), pp. 105. http://fs.gallup.unm.edu/eBook-neutrosophic (fourth version)
27. Tanay, B; Kandemir, M. B. Topological structure of fuzzy soft sets., Computers & Mathematics with Applications, (2011), 61(10), pp. 2952—2957.
28. Topal, S.; Broumi, S.; Bakali, A.; Talea, M.; Smarandache, F. A python tool for implementations on bipolar neutrosophic matrices, Neutrosophic Sets and Systems, (2019), 28, pp. 138–161. 10.5281/zenodo.3382529
29. Zadeh, L. A. Fuzzy sets, Information and Control, (1965), 8(3), pp. 338—353.
30. Zahariev, Z. Software package and API in MATLAB for working with fuzzy algebras. In G. Venkov, R. Kovacheva, & V. Pasheva (Eds.), AIP Conference Proceedings, (2009), 1184(1), pp. 341–348.

Received: July 10, 2023. Accepted: Nov 24, 2023