SoftwareX 12 (2020) 100632

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

PyIVNS: A python based tool for Interval-valued neutrosophic operations and normalization

Ahmed Sleem *, Mohamed Abdel-Baset, Ibrahim El-henawy

Faculty of Computers and Informatics, Zagazig University, Sharqiyah, Egypt

ARTICLE INFO

Article history:
Received 30 January 2020
Received in revised form 8 June 2020
Accepted 17 November 2020

Keywords:
Python
Neutrosophic
Normalization

ABSTRACT

A lot of research has been proposed to solve neutrosophic problems. The proposed algorithms and solutions misses of basic operations calculations tool to build on it. IVns is sharing a good percentage of those proposals and preferred in many situations. PyIVNS is a python tool to allow researchers to perform operations on interval-valued neutrosophic sets (IVNS). Operations can be done over matrices of IVNS. Also, PyIVNS tool can perform normalization for matrices using several methods of normalization (Linear, Linear by min–max, linear by sum, vector, and enhanced accuracy). PyIVNS Tool can be embedded in other software or tool or it can be used via its web application.

Code metadata

Current code version	*V1.0*
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_18
Code Ocean compute capsule	
Legal Code License	*BSD-3-Clause*
Code versioning system used	*git*
Software code languages, tools, and services used	*Python3.6, Django2.5*
Compilation requirements, operating environments & dependencies	*Linux, web server, python 3.6, Django 2.5*
If available Link to developer documentation/manual	
Support email for questions	Ahmedsleem8000@gmail.com

Software metadata

Current software version	*V1.0*
Permanent link to executables of this version	http://ahmedsleem.pythonanywhere.com/ivns/
Legal Software License	*BSD-3-Clause*
Computing platforms/Operating Systems	*Windows/Linux/mac*
Installation requirements & dependencies	*Web browser for example, chrome, Firefox No installation for client side*
If available, link to user manual - if formally published include a reference to the publication in the reference list	
Support email for questions	Ahmedsleem8000@gmail.com

1. Motivation and significance

The problem of uncertainty was taking the concern researches a long time ago. Fuzzy sets theory proposed in [1] to present solution using the membership function. A lot of researchers are using fuzzy and many applications depended on fuzzy set theory to build new algorithms and solution for real world problems. Some of those real world problems cannot be expressed using one single value in membership function, so Interval valued fuzzy sets were proposed in [2]. Intuitionistic fuzzy sets introduced later to present the none-membership degree [3]. And of course, intuitionistic fuzzy sets facing difficulties in some cases to determine single value for membership degree and none-membership degree. So, interval intuitionistic fuzzy sets proposed in [4] to handle the hesitance of determining such degrees.

While it seems to the world that nothing is better, Smarandache introduce the philosophy of Neutrosophic sets theory [5].

* Corresponding author.
E-mail addresses: ahmedsleem8000@gmail.com (A. Sleem), analyst_mohamed@yahoo.com (M. Abdel-Baset), henawy2000@yahoo.com (I. El-henawy).

https://doi.org/10.1016/j.softx.2020.100632

In [5–8], he introduces Neutrosophic sets as an alternative of fuzzy sets theory to cope with problems that has indeterminacy degree which is independent of membership degree and none-membership degree. Neutrosophic sets expressed as a degree of truth, indeterminacy, and falsity which represent the degree of membership, and indeterminacy, and none-membership respectively. It was difficult to use NS in real applications until SVNS (single-valued neutrosophic set) has been proposed in [9]. SVNS opened the field for introducing many researches works on neutrosophic operations and aggregations [10–17]. SVNS value expressed as x(T,I,F) where T,I,F are Truth membership degree , indeterminacy membership degree, and falsity membership degree respectively .

For example: x(0.5,0.2,0.6) to show truth membership is 0.5, indeterminacy membership is 0.2, and falsity membership is 0.6.

But, this seems insufficient for real world problems, which was in need to IVNS (Interval-valued neutrosophic sets). IVNS introduced in [18] to define the truth membership, indeterminacy membership, and falsity membership as an interval instead of single value. IVNS expressed as x < (TL, TU), (IL, IU), (FL, FU)> where TL is truth lower value, TU truth upper value, IL is indeterminacy lower value, IU indeterminacy upper value, FL falsity lower value, and FU falsity upper value of interval-valued neutrosophic number, for example : x < (0.4,0.7), (0.1,0.3), (0.2,0.5)>.

In [19], Smarandache proved that neutrosophic set is a generalization of intuitionistic fuzzy and inconsistent intuitionistic fuzzy sets. The author compared between the neutrosophic operators and fuzzy operators applied to same problems. He cleared that two main reasons give advantage to neutrosophic sets: *first*, the **independence** of the truth-membership, falsity-membership, and indeterminacy-membership functions. *Second*, and the role of **indeterminacy** membership function that is not exist in all previous set theories. That clarification show the benefits of using neutrosophic operators in software programs rather than previous sets operators.

In [20,21], the authors presented a Matlab tool for basic operations on single-valued and interval-valued neutrosophic sets. They proposed matrices operations with clear difficulties in get inputs. It considered good seed to perform basic operations on neutrosophic sets but It is not easy to embed this tool in external software in Matlab while it is not possible to use it outside Matlap projects.

A python tool proposed in [22] to perform operations on bipolar neutrosophic matrices using Numby module. They used nested list data types to simulate matrices and it is good base to build on it.

This paper inspired from basic preliminaries and definitions of interval-valued neutrosophic set as we will demonstrate some of these preliminaries.

Definition 1 (*[18], Interval-Valued Neutrosophic Set*)**.** Let X be a space of points (objects) with a generic element in X denoted by x. A neutrosophic set A in X characterized by a truth-membership function TA, an indeterminacy-membership function IA and a falsity-membership function FA. TA, IA and FA are real standard or non-standard subsets of]0−, 1+[.

$x_i = \langle [T_i^L, T_i^U], [I_i^L, I_i^U], [F_i^L, F_i^U] \rangle$ is a collection of interval-valued neutrosophic numbers where i = 1,2,.....n and n is the number of decision makers.

Definition 2 (*[18], Addition*)**.** The addition of two interval neutrosophic sets A and B is an interval neutrosophic set C, written as C = A + B, whose truth-membership, indeterminacy-membership and false-membership functions are related to those of A and B by

$$T_C^L(x) = \min\left(T_A^L(x) + T_B^L(x), 1\right)$$
$$T_C^U(x) = \min\left(T_A^U(x) + T_B^U(x), 1\right)$$
$$I_C^L(x) = \min\left(I_A^L(x) + I_B^L(x), 1\right)$$
$$I_C^U(x) = \min\left(I_A^U(x) + I_B^U(x), 1\right)$$
$$F_C^L(x) = \min\left(F_A^L(x) + F_B^L(x), 1\right)$$
$$F_C^U(x) = \min\left(F_A^U(x) + F_B^U(x), 1\right)$$

for all x in X.

Definition 3 (*[18], Difference*)**.** The difference of two interval neutrosophic sets A and B is an interval neutrosophic set C, written as C = A \B, whose Truth-membership, indeterminacy-membership and false-membership functions are related to those of A and B by

$$T_C^L(x) = \min\left(T_A^L(x), F_B^L(x)\right)$$
$$T_C^U(x) = \min\left(T_A^U(x), F_B^U(x)\right)$$
$$I_C^L(x) = \max\left(I_A^L(x), 1 - I_B^U(x)\right)$$
$$I_C^U(x) = \max\left(I_A^U(x), 1 - I_B^L(x)\right)$$
$$F_C^L(x) = \max\left(F_A^L(x), T_B^L(x)\right)$$
$$F_C^U(x) = \max\left(F_A^U(x), T_B^U(x)\right)$$

For all x in X.

Definition 4 (*[18], Scalar Multiplication*)**.** The scalar multiplication of interval neutrosophic set A is an interval neutrosophic set B, written as B = a.A, whose truth-membership, indeterminacy-membership and false-membership functions are related to those of A by

$$T_{B|}^L(x) = \min\left(T_A^L(x).a, 1\right)$$
$$T_B^U(x) = \min\left(T_A^U(x).a, 1\right)$$
$$I_B^L(x) = \min\left(I_A^L(x).a, 1\right)$$
$$I_B^U(x) = \min\left(I_A^U(x).a, 1\right)$$
$$F_B^L(x) = \min\left(F_A^L(x).a, 1\right)$$
$$F_B^U(x) = \min\left(F_A^U(x).a, 1\right)$$

For all x in X.

Definition 5 (*[18], Cartesian Product*)**.** The Cartesian product of two interval neutrosophic sets A defined on universe X and B defined on universe Y is an interval neutrosophic set C, written as C = A × B, whose truth-membership, indeterminacy-membership and false-membership functions are related to those of A and B by

$$T_C^L(x, y) = T_A^L(x) + T_B^L(y) - T_A^L(x).T_B^L(y)$$
$$T_C^U(x, y) = T_A^U(x) + T_B^U(y) - T_A^U(x).T_B^U(y)$$
$$I_C^L(x, y) = I_A^L(x).I_B^L(y)$$
$$I_C^U(x, y) = I_A^U(x).I_B^U(y)$$
$$F_C^L(x, y) = F_A^L(x).F_B^L(y)$$
$$F_C^U(x, y) = F_A^U(x).F_B^U(y)$$

For all x in X, y in Y.

Definition 6 (*[23], Normalization*)**.** For m × n matrix, each element in the matrix (x_{ij}) is IVNS value in format <[TL, TU], [IL, IU], [FL, FU]>

The normalization types defined in following definitions [ ].

Definition 6.1 (*Linear Normalization*)**.** For beneficial neutrosophic value: $\overline{x_{ij}} = \frac{x_{ij}}{x_j^{max}}$

For non-beneficial neutrosophic value: $\overline{x_{ij}} = 1 - \frac{x_{ij}}{x_j^{max}}$

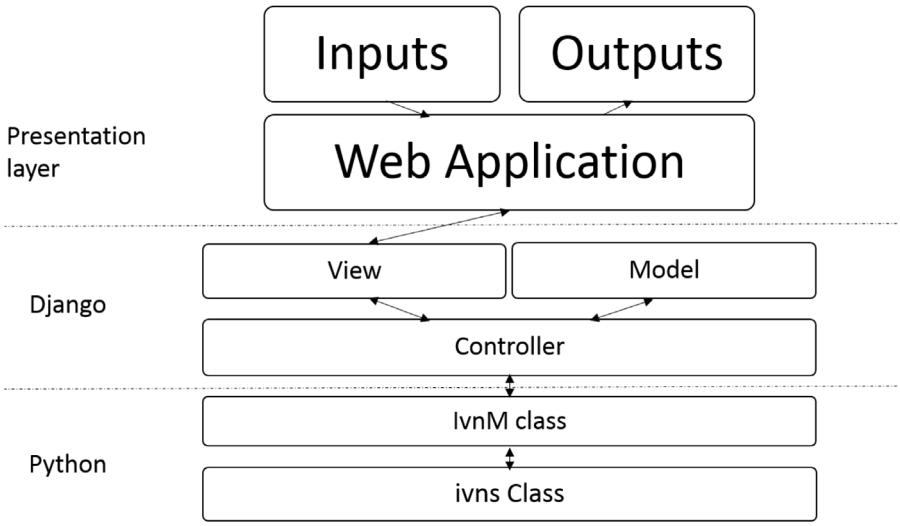

Fig. 1. Software hierarchy.

Definition 6.2 (*Linear Normalization (Max–Min)*)**.** For beneficial neutrosophic value: $\overline{x_{ij}} = \frac{x_{ij}-x_j^{min}}{x_j^{max}-x_j^{min}}$

For non-beneficial neutrosophic value: $\overline{x_{ij}} = 1 - \frac{x_j^{max}-x_{ij}}{x_j^{max}-x_j^{min}}$

Definition 6.3 (*Linear Normalization (Sum)*)**.** For beneficial neutrosophic value: $\overline{x_{ij}} = \frac{x_{ij}}{\sum_{j=1}^{m} x_{ij}}$

For non-beneficial neutrosophic value: $\overline{x_{ij}} = 1 - \frac{1/x_{ij}}{\sum_{j=1}^{m}(1/x_{ij})}$

Definition 6.4 (*Vector Normalization*)**.** For beneficial neutrosophic value: $\overline{x_{ij}} = \frac{x_{ij}}{\sqrt{\sum_{j=1}^{m} x_{ij}^2}}$

For non-beneficial neutrosophic value: $\overline{x_{ij}} = 1 - \frac{x_{ij}}{\sqrt{\sum_{j=1}^{m} x_{ij}^2}}$

Definition 6.5 (*Enhanced Accuracy Normalization*)**.** For beneficial neutrosophic value: $\overline{x_{ij}} = 1 - \frac{x_j^{max}-x_{ij}}{\sum_{j=1}^{m}(x_j^{max}-x_{ij})}$

For non-beneficial neutrosophic value: $\overline{x_{ij}} = 1 - \frac{x_{ij}-x_j^{min}}{\sum_{j=1}^{m}(x_{ij}-x_j^{min})}$

2. Software description

Proposed software work with Interval valued neutrosophic sets (IVNS) to make it easy to be handled and calculated. The presented software consist of 2 parts. First: it performs operation on IVNS. Second: it performs normalization for IVNS matrices. The software allow inputs in form of matrix or single IVNS number. It also has been implemented as a web application interface. It has been coded in python 3 as a backend and using Django framework for front end. Django is an open source web framework based python.

Inputs can be entered or pasted from data file in specified format as explained in the main screen.

The result in the same format and can be exported by copying it from the result text area.

2.1. Software architecture

The software implemented as a web application interface. It is coded in python 3 as a backend and using Django framework for front end. Django is an open source web framework based python.

Fig. 1 illustrate the hierarchy of the software and the boundaries of implementation tools.

Also it shows the main classes and their relations with the web application.

The base class which is make the calculations on the level of neutrosophic number is ivns class. This class is carrying the interval valued neutrosophic number as attributes called TL, TU, IL, IU, FL, and FU to represent the lower and upper truth-membership function, lower and upper indeterminacy-membership function, lower and upper falsity-membership function.

IvnM is another python class that perform the calculations on matrix of ivns numbers.

The user enter his input (ivns matrix) throw the presentation layer and system pass it to the controller to select the appropriate function from IvnM class to calculate the result.

The IvmM class create the result matrix by calling ivnm class for each matrix element then respond the resulted matrix to the controller which passes the result to View Class to handle output resulted matrix to the user. Of course, validation has been done to ensure inputs are in the specified format and each matrix element represent ivns number.

2.2. Software functionalities

There are two main functions illustrated in the top menu of the software.

1. IVNS operation
2. Neutrosophic matrix normalization

The use case diagram for IVNS operations are illustrated in Fig. 2. It shows the list of calculations that user can get from the system on level of single IVNS number or level of matrix of IVNS numbers. Fig. 3 illustrate the use case diagram for IVNS normalization and it shows types of normalization that user can calculate. Always user can copy the selected result to use it easily. Screen shots from the running web application showed in Figs. 4 and 5. IVNS operation page illustrated in Fig. 4 with result for intersection between matrix A and matrix B. while Fig. 5 illustrate the result of normalizing matrix of type beneficial with linear normalization

Ivns operations contain the following functions:

- Calculate complement, product scalar, power scalar of ivns matrix.
- Calculate Karasan score, Ridvan score, and Nancy score for ivns matrix.

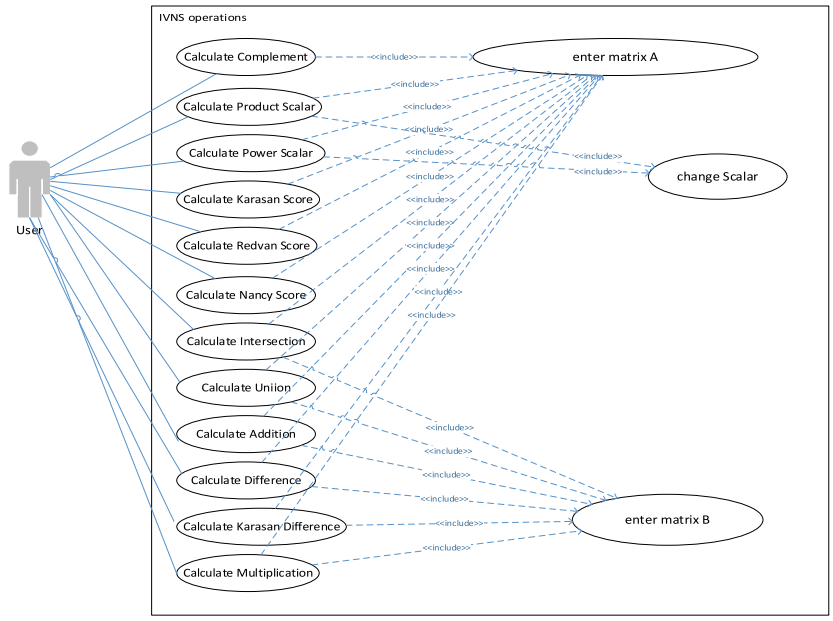

Fig. 2. Use case diagram for IVNS operations.

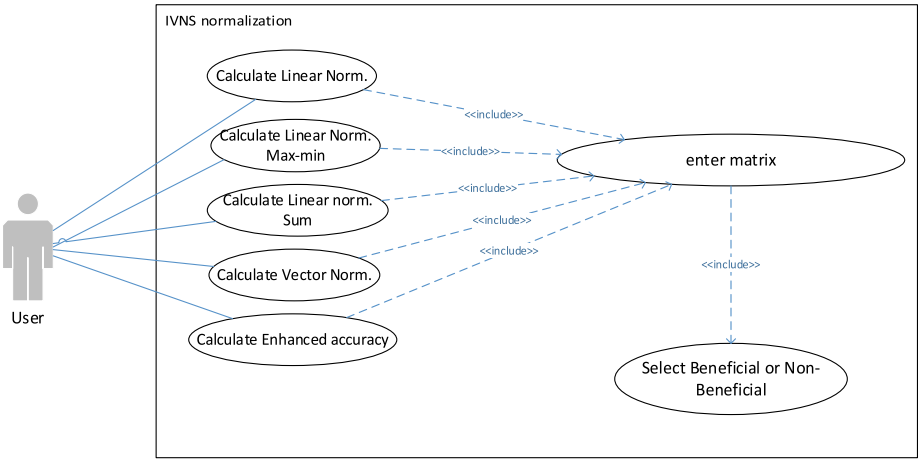

Fig. 3. Use case diagram for IVNS Normalization.

- Calculate the intersection, union, addition, difference, multiplication of two ivns matrices.

Neutrosophic matrix normalization performs following functions for beneficial and non-beneficial matrix:

- Linear normalization.
- Linear normalization (maximum and minimum)
- Linear normalization (Summation)
- Vector Normalization
- Enhanced Accuracy Normalization.

2.3. Sample code snippets analysis

The software has 2 main classes Ivns class and IvnM class. The first class is responsible of carrying the IVNS attributes TL, TU, IL, IU, FL, and FU. Also it has functions to perform the operations on the level of IVNS number. While second class is responsible of IVNS matrix calculations, creation of matrix from user input text, validating the matrix format and size, and outputting the result in appropriate format. Also, class IvnM contains the related normalization functions considering that normalization is a matrix operation. Following is a snippet code example for function Union from class Ivns.

```
def Union(self,a,b):
    c = Ivns(0, 0, 0, 0, 0, 0, )
    c.tl = max(float(a.tl), float(b.tl))
    c.tu = max(float(a.tu), float(b.tu))
    c.il = min(float(a.il), float(b.il))
    c.iu = min(float(a.iu), float(b.iu))
    c.fl = min(float(a.fl), float(b.fl))
    c.fu = min(float(a.fu), float(b.fu))

    return c
```

From class IvnM we illustrate the function called “Create” which is responsible of converting the user input text into matrix of type IVNS.

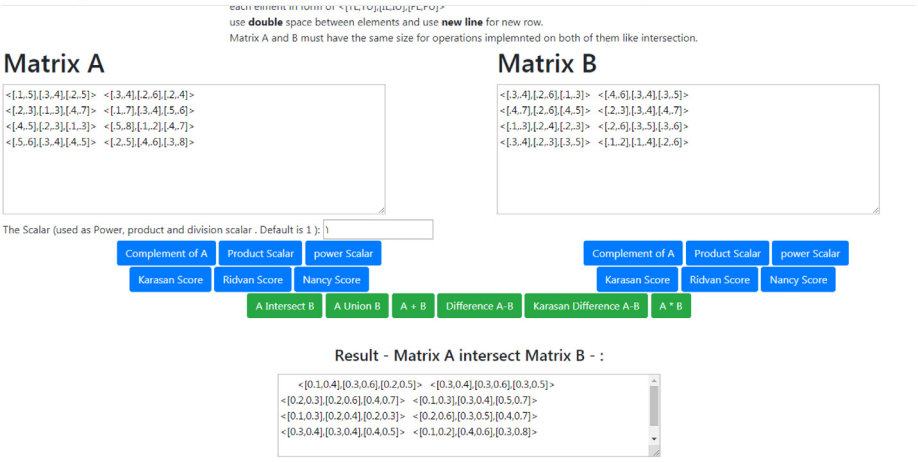

Fig. 4. User interface for neutrosophic operations page.

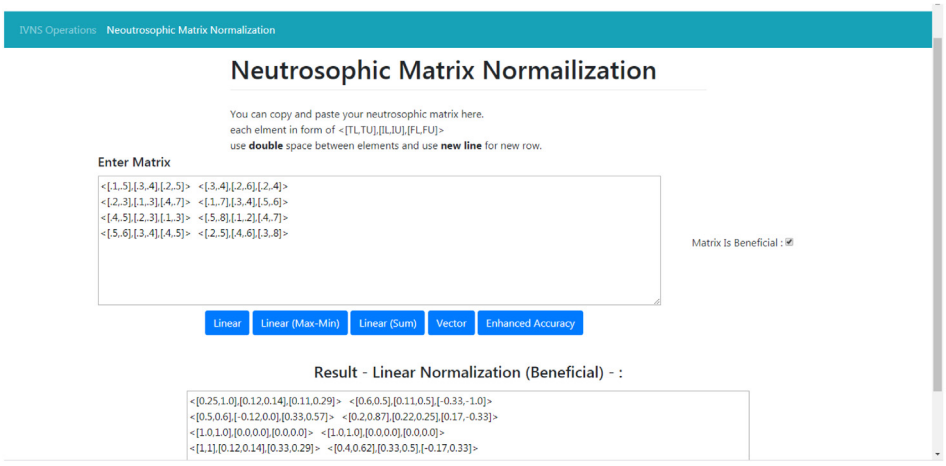

Fig. 5. User interface for normalization page.

Table 1
Matrix A and B.

Matrix A	Matrix B
<[.1,.5],[.3,.4],[.2,.5]> <[.3,.4],[.2,.6],[.2,.4]>	<[.3,.4],[.2,.6],[.1,.3]> <[.4,.6],[.3,.4],[.3,.5]>
<[.2,.3],[.1,.3],[.4,.7]> <[.1,.7],[.3,.4],[.5,.6]>	<[.4,.7],[.2,.6],[.4,.5]> <[.2,.3],[.3,.4],[.4,.7]>
<[.4,.5],[.2,.3],[.1,.3]> <[.5,.8],[.1,.2],[.4,.7]>	<[.1,.3],[.2,.4],[.2,.3]> <[.2,.6],[.3,.5],[.3,.6]>
<[.5,.6],[.3,.4],[.4,.5]> <[.2,.5],[.4,.6],[.3,.8]>	<[.3,.4],[.2,.3],[.3,.5]> <[.1,.2],[.1,.4],[.2,.6]>

```
def Create(self):
    matrix = []
    text=self.text.split('\n')
    for line in text:
        items = line.split('  ')
        row=[]
        for item in items:
            item =item.translate({ord(i): None for i in
'<>[]'})
            item= item.translate({ord(i): None for i in ' '})
            numbers=item.split(',')

ivns=Ivns(numbers[0],numbers[1],numbers[2],numbers[3],numbers
[4],numbers[5],)
            row.append(ivns)
        matrix.append(row)
    return matrix
```

Python classes can be used without the web application and these classes can be embedded in any other tool or software and it can be extended to perform further specified calculations.

3. Illustrative examples

3.1. IVNS operations example

To illustrate the system we use matrix A and matrix B as in Table 1. They contains random IVNS numbers.

We use the online running web application http://ahmedsleem.pythonanywhere.com/ivns/ to demonstrate the proposed example. And we used scalar integer $S=2$. The results calculated and collected to be showed in Table 2.

The results have been tested and compared with the manual solution. W make a detailed tracing to make sure the accuracy of results. So, this example can be used as reference for any other tool.

3.2. IVNS normalization example

In this example we used the online running web application http://ahmedsleem.pythonanywhere.com/ivns/ to get the result. The software used formulas defined in Definition 6 from

Table 2
Results for IVNS operations on matrix A and B.

F	Result	F	Result
Complement of A	<[0.2,0.5],[0.6,0.7],[0.1,0.5]> <[0.2,0.4],[0.4,0.8],[0.3,0.4]> <[0.4,0.7],[0.7,0.9],[0.2,0.3]> <[0.5,0.6],[0.6,0.7],[0.1,0.7]> <[0.1,0.3],[0.7,0.8],[0.4,0.5]> <[0.4,0.7],[0.8,0.9],[0.5,0.8]> <[0.4,0.5],[0.6,0.7],[0.5,0.6]> <[0.3,0.8],[0.4,0.6],[0.2,0.5]>	A intersect B	<[0.1,0.4],[0.3,0.6],[0.2,0.5]> <[0.3,0.4],[0.3,0.6],[0.3,0.5]> <[0.2,0.3],[0.2,0.6],[0.4,0.7]> <[0.1,0.3],[0.3,0.4],[0.5,0.7]> <[0.1,0.3],[0.2,0.4],[0.2,0.3]> <[0.2,0.6],[0.3,0.5],[0.4,0.7]> <[0.3,0.4],[0.3,0.4],[0.4,0.5]> <[0.1,0.2],[0.4,0.6],[0.3,0.8]>
Product scalar A *2	<[0.2,1.0],[0.6,0.8],[0.4,1.0]> <[0.6,0.8],[0.4,1],[0.4,0.8]> <[0.4,0.6],[0.2,0.6],[0.8,1]> <[0.2,1],[0.6,0.8],[1.0,1]> <[0.8,1.0],[0.4,0.6],[0.2,0.6]> <[1.0,1],[0.2,0.4],[0.8,1]> <[1.0,1],[0.6,0.8],[0.8,1.0]> <[0.4,1.0],[0.8,1],[0.6,1]>	A Union B	<[0.3,0.5],[0.2,0.4],[0.1,0.3]> <[0.4,0.6],[0.2,0.4],[0.2,0.4]> <[0.4,0.7],[0.1,0.3],[0.4,0.5]> <[0.2,0.7],[0.3,0.4],[0.4,0.6]> <[0.4,0.5],[0.2,0.3],[0.1,0.3]> <[0.5,0.8],[0.1,0.2],[0.3,0.6]> <[0.5,0.6],[0.2,0.3],[0.3,0.5]> <[0.2,0.5],[0.1,0.4],[0.2,0.6]>
Power scalar A^2	<[0.01,0.25],[0.09,0.16],[0.04,0.25]> <[0.09,0.16],[0.04,0.36],[0.04,0.16]> <[0.04,0.09],[0.01,0.09],[0.16,0.49]> <[0.01,0.49],[0.09,0.16],[0.25,0.36]> <[0.16,0.25],[0.04,0.09],[0.01,0.09]> <[0.25,0.64],[0.01,0.04],[0.16,0.49]> <[0.25,0.36],[0.09,0.16],[0.16,0.25]> <[0.04,0.25],[0.16,0.36],[0.09,0.64]>	A + B	<[0.37,0.7],[0.06,0.24],[0.02,0.15]> <[0.58,0.76],[0.06,0.24],[0.06,0.2]> <[0.52,0.79],[0.02,0.18],[0.16,0.35]> <[0.28,0.79],[0.09,0.16],[0.2,0.42]> <[0.46,0.65],[0.04,0.12],[0.02,0.09]> <[0.6,0.92],[0.03,0.1],[0.12,0.42]> <[0.65,0.76],[0.06,0.12],[0.12,0.25]> <[0.28,0.6],[0.04,0.24],[0.06,0.48]>
Karasan score of A	0.41166666666666657 0.4200000000000001 0.37333333333333324 0.3683333333333333 0.625 0.6233333333333334 0.4766666666666666 0.26666666666666666	A – B	<[.3,.4],[.2,.6],[.1,.3]> <[.4,.6],[.3,.4],[.3,.5]> <[.4,.7],[.2,.6],[.4,.5]> <[.2,.3],[.3,.4],[.4,.7]> <[.1,.3],[.2,.4],[.2,.3]> <[.2,.6],[.3,.5],[.3,.6]> <[.3,.4],[.2,.3],[.3,.5]> <[.1,.2],[.1,.4],[.2,.6]>
Ridvan score of A	0.5333333333333333 0.5499999999999999 0.5000000000000001 0.49999999999999994 0.6666666666666669 0.6499999999999999 0.5833333333333333 0.4333333333333333	A-B (karasan)	<[−0.2,0.4],[0.3,0.6],[−0.2,0.2]> <[−0.2,0.1],[0.3,0.6],[−0.4,0.0]> <[−0.3,−0.1],[0.2,0.6],[−0.3,0.3]> <[−0.6,0.3],[0.3,0.4],[0.2,0.4]> <[0.1,0.3],[0.2,0.4],[−0.2,0.2]> <[−0.1,0.5],[0.3,0.5],[−0.2,0.5]> <[0.0,0.3],[0.3,0.4],[0.0,0.2]> <[−0.4,0.3],[0.4,0.6],[0.1,0.7]>
Nancy score of A	−0.1937500000000001 −0.1562500000000001 −0.02499999999999991 −0.24375000000000002 0.26875000000000004 0.34 0.020000000000000073 −0.4600000000000002	A * B	<[0.03,0.2],[0.44,0.76],[0.28,0.65]> <[0.12,0.24],[0.44,0.76],[0.44,0.7]> <[0.08,0.21],[0.28,0.72],[0.64,0.85]> <[0.02,0.21],[0.51,0.64],[0.7,0.88]> <[0.04,0.15],[0.36,0.58],[0.28,0.51]> <[0.1,0.48],[0.37,0.6],[0.58,0.88]> <[0.15,0.24],[0.44,0.58],[0.58,0.75]> <[0.02,0.1],[0.46,0.76],[0.44,0.92]>

Table 3
Input matrix used in IVNS normalization.

Input matrix
<[.1,.5],[.3,.4],[.2,.5]> <[.3,.4],[.2,.6],[.2,.4]>
<[.2,.3],[.1,.3],[.4,.7]> <[.1,.7],[.3,.4],[.5,.6]>
<[.4,.5],[.2,.3],[.1,.3]> <[.5,.8],[.1,.2],[.4,.7]>
<[.5,.6],[.3,.4],[.4,.5]> <[.2,.5],[.4,.6],[.3,.8]>

first section of this paper. We will use the matrix given in Table 3 to illustrate the resulted IVNS normalization over specified normalization types.

The resulted normalized matrix illustrated in Table 4 beside it is normalization type.in this example we assume that matrix is beneficial.

This example matrix solved manually before it used in PyIVNS tool. So, we make sure from the results. Later, we can make the result shown in steps but now at least we are sure from our results.

4. Impact

Many solutions can be built on the proposed work such as neutrosophic AHP algorithms because introduced software considered as a base for iVNS neutrosophic operations calculations and present a way to researchers to use it inside their software. Also, the can benefit from the tool interface to get quick results that help them test their proposals algorithms and solutions.

Beside it is considered a great tool for students to deal with neutrosophic numbers to help them imagine and feel philosophy behind it.

PyIVNS tool Will encourage researchers to build tools for their theoretical work as they will focus on their proposal rather than stick in difficult calculations which is out of their scope in many situations.

The presented web application itself considered as an example of using PyIVNS tool inside software. The web application embed the PyIVNS classes to allow users make their calculations. The classes can embedded as open source code or as black box classes. In the future, developing API (Application Programming Interface) version will be helpful in using the tool as a service.

5. Conclusions

PyIVNS tool has been presented in this paper to perform interval-valued neutrosophic operations and normalization and we show how researchers can be benefit from this tools. PyIVNS can be extended to present neutrosophic algorithms such as AHP algorithms, TOPISS algorithms and further more. This tool

Table 4
IVNS normalization results.

Type	Result
Linear	<[0.25,1.0],[0.12,0.14],[0.11,0.29]> <[0.6,0.5],[0.11,0.5],[−0.33,−1.0]> <[0.5,0.6],[−0.12,0.0],[0.33,0.57]> <[0.2,0.87],[0.22,0.25],[0.17,−0.33]> <[1.0,1.0],[0.0,0.0],[0.0,0.0]> <[1.0,1.0],[0.0,0.0],[0.0,0.0]> <[1,1],[0.12,0.14],[0.33,0.29]> <[0.4,0.62],[0.33,0.5],[−0.17,0.33]>
Linear (Max-Min)	<[1.0,1.0],[0.0,0.0],[0.0,0.0]> <[0.75,0.57],[0.0,0.0],[0.0,0.0]> <[1.0,1.0],[0.0,0.0],[0.0,0.4]> <[0.25,1.0],[0.0,0.0],[0.0,0.0]> <[1.0,1.0],[0.0,0.0],[0.0,0.0]> <[1.0,1.0],[0.0,0.0],[0.0,0.0]> <[1.0,1.0],[0.0,0.0],[0.0,0.0]> <[0.5,0.71],[0.0,0.0],[0.0,0.0]>
Linear (Sum)	<[0.13,0.54],[0.3,0.39],[0.2,0.47]> <[0.4,0.41],[0.2,0.59],[0.19,0.3]> <[0.26,0.32],[0.1,0.29],[0.4,0.68]> <[0.13,0.71],[0.3,0.38],[0.49,0.53]> <[0.51,0.54],[0.2,0.29],[0.1,0.26]> <[0.67,0.82],[0.1,0.18],[0.39,0.65]> <[0.64,0.65],[0.3,0.39],[0.4,0.47]> <[0.27,0.51],[0.4,0.59],[0.29,0.77]>
Vector	<[0.16,0.61],[0.3,0.4],[0.2,0.5]> <[0.51,0.43],[0.2,0.6],[0.2,0.3]> <[0.32,0.37],[0.1,0.3],[0.4,0.7]> <[0.17,0.74],[0.3,0.4],[0.5,0.53]> <[0.63,0.61],[0.2,0.3],[0.1,0.3]> <[0.85,0.85],[0.1,0.2],[0.4,0.65]> <[0.79,0.73],[0.3,0.4],[0.4,0.5]> <[0.34,0.53],[0.4,0.6],[0.3,0.77]>
Enhanced accuracy	<[0.09,0.48],[0.46,0.48],[0.27,0.54]> <[0.38,0.59],[0.29,0.65],[0.24,0.4]> <[0.19,0.27],[0.15,0.36],[0.54,0.54]> <[0.38,0.59],[0.43,0.43],[0.6,0.61]> <[0.39,0.48],[0.31,0.36],[0.14,0.33]> <[0.48,0.73],[0.14,0.22],[0.48,0.71]> <[0.49,0.58],[0.46,0.48],[0.54,0.54]> <[0.38,0.59],[0.57,0.65],[0.36,0.81]>

contains the basic operations needed for interval-valued neutrosophic sets and assumed to make developing neutrosophic solutions and algorithms easier. Future work can be made for other types of neutrosophic numbers such as single-valued, Triangular, and Trapezoidal neutrosophic numbers.

Future directions will be building API (Application Programming Interface) version to allow researchers easily embed the tool in their software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors are very grateful to the chief editor and reviewers for their comments and suggestions, which is helpful in improving the paper.

References

[1] Zadeh LA. Fuzzy sets. Inf Control 1965;8:338–53.
[2] Gorzalczany MB. A method of inference in approximate reasoning based on interval-valued fuzzy sets. Fuzzy Sets and Systems 1987;21:1–17.
[3] Atanassov KT. Intuitionistic fuzzy sets. Fuzzy Sets and Systems 1986;20(1):87–96.
[4] Atanassov K, Gargov G. Interval valued intuitionistic fuzzy sets. Fuzzy Sets and Systems 1989;31(3):343–9.
[5] Smarandache F. A Unifying Field in Logics. Neutrosophy: Neutrosophic Probability, Set and Logic. Rehoboth, Mass, USA: American Research Press; 1999.
[6] Smarandache F. A Unifying Field in Logics Neutrosophic Logic. Neutrosophy, Neutrosophic Set, Neutrosophic Probability. American Research Press; 2003.
[7] Rivieccio U. Neutrosophic logics: prospects and problems. Fuzzy Sets and Systems 2008;159(14):1860–8.
[8] Majumdar P, Samant SK. On similarity and entropy of neutrosophic sets. J Intell Fuzzy Systems 2014;26(3):1245–52.
[9] Wang H, Smarandache F, Zhang YQ, Sunderraman R. Single valued neutrosophic sets. Multispace Multistructure 2010;4:410–3.
[10] Ye J. A multicriteria decision-making method using aggregation operators for simplified neutrosophic sets. J Intell Fuzzy Systems 2013.
[11] Ye J. Multicriteria decision-making method using the correlation coefficient under single-value neutrosophic environment. Int J Gen Syst 2013;42(4):386–94.
[12] Ye J. Multiple attribute group decision-making method with completely unknown weights based on similarity measures under single valued neutrosophic environment. J Intell Fuzzy Systems 2014;27:2927–35.
[13] Ye J. Multicriteria decision-making method using the correlation coefficient under single-valued neutrosophic environment. Int J Gen Syst 2013;42:386–94.
[14] Ye J. Improved correlation coefficients of single valued neutrosophic sets and interval neutrosophic sets for multiple attribute decision making. J Intell Fuzzy Systems 2014;27:2453–62.
[15] Ye J. Clustering methods using distance-based similarity measures of single-valued neutrosophic sets. J Intell Fuzzy Systems 2014;23:379–89.
[16] Huang HL. New distance measure of single-valued neutrosophic sets and its application. Int J Intell Syst 2016;31(10):1021–32.
[17] Yang HL, Guo ZL, She YH, Liao XW. On single valued neutrosophic relations. J Intell Fuzzy Systems 2016;30:1045–56.
[18] Wang H. Interval neutrosophic sets and logic: theory and applications in computing. Comput Sci 2005;65(4):1–99.
[19] Smarandache F. Neutrosophic set is a generalization of intuitionistic fuzzy set, inconsistent intuitionistic fuzzy set (picture fuzzy set, ternary fuzzy set), pythagorean fuzzy set (Atanassov's intuitionistic fuzzy set of second type), q-Rung orthopair fuzzy set, spherical fuzzy set, and n-HyperSpherical fuzzy set, while neutrosophication is a generalization of regret theory, grey system theory, and three-ways decision (revisited). J New Theory 2019;29:01–31.
[20] Broumi S, Son LH, Bakali A, Talea M, Smarandache F, Selvachandran G. Computing operational matrices in neutrosophic environments: A Matlab toolbox. Neutrosophic Sets and Systems 2017;18:58–66.
[21] Broumi S, Bakali A, Talea M, Smarandache F. A Matlab toolbox for interval valued neutrosophic matrices for computer applications. Uluslararası Yönetim Bilişim Sistemlerive Bilgisayar Bilimleri Dergisi 2017;1(1):1–21.
[22] Topal S, Broumi S, Bakali A, Talea M, Smarandache F. A python tool for implementations on bipolar neutrosophic matrices. Neutrosophic Sets and Systems 2019;28:138–61. http://dx.doi.org/10.5281/zenodo.3382529.
[23] Tian Zhang-peng, Wang Jing, Zhang H-Y, Chen Xiao-hong, Wang Jianqiang. Simplified neutrosophic linguistic normalized weighted Bonferroni mean operator and its application to multi-criteria decision-making problems. Filomat 2016;30:3339–60. http://dx.doi.org/10.2298/FIL1508576F.