Notes on Intuitionistic Fuzzy Sets
Print ISSN 1310–4926, Online ISSN 2367–8283
2022, Volume 28, Number 1, Pages 51–85
DOI: 10.7546/nifs.2022.28.1.51-85

Software implementation of intuitionistic fuzzy sets and some operators

Evgeniy Marinov[1,2]

[1] Department of Bioinformatics and Mathematical Modelling,
Institute of Biophysics and Biomedical Engineering,
Bulgarian Academy of Sciences
105 Acad. G. Bonchev Str., 1113 Sofia, Bulgaria

[2] Gate Institute, Sofia University "St. Kliment Ohridski"
125 Tsarigradsko Shose Blvd., Bl. 2
1113 Sofia, Bulgaria
e-mail: evgeniy.iv.marinov@gmail.com

Received: 25 February 2022 **Revised:** 28 March 2022 **Accepted:** 1 April 2022

Abstract: In this paper, we present a software implementation of the framework of Intuitionistic Fuzzy Sets (IFSs). The presented implementation allows the user to interactively shape an IFS, to compute, plot and visualize various of operators for IFS and allows for the modeling of real world problems.
Keywords: Intuitionistic fuzzy set, Intuitionistic fuzzy interpretational triangle, Python, Software implementation.
2020 Mathematics Subject Classification: 03E72.

1 Introduction

The Intuitionistic Fuzzy Sets (IFSs) were introduced in 1983 in [2]. Its theory was described in details in [4,9], but by the moment there is only one attempt for software implementation of some IFS-operations and operators [1].

The proposed in this paper software implementation for visualization and computation of IFSs, and operations and operators over those, can be applied in the modeling of real world problems. We used Python as programming language. In addition to that, we implement a

functionality for serialization of the IFSs in json files. The serialized IFSs can have properties, such as size of the circle in the triangle representation, colour, etc.

2 Short remarks on IFSs

Since the introduction of fuzzy sets by Zadeh [20] there have been a number of generalizations. Most of them consist of replacing the range $[0, 1]$ by more general algebraic structures satisfying the axioms for a lattice (cf. Birkhoff [13]) – they are called *L-fuzzy sets* (cf. Goguen [14]).

A very popular extension of fuzzy sets is the Atanassov's generalization - *intuitionistic fuzzy* set (IFS) (cf. Atanassov [2, 4, 9]), where the corresponding lattice takes the natural form of a triangular representation (described in the next section). In addition to the membership function of a FS, there is another function, expressing a notion of non-membership degree with the same domain X and range $[0, 1]$, for which the sum of the membership and non-membership degrees should never exceed 1. That is, in the framework of IFSs we have an additional degree, expressing the lack of knowledge/information. This makes the theory invaluable to extend the uncertainty of the limited level of crisp and even fuzzy precision in real world situations and preferences.

In this section, following [4, 9], we will give the basic concepts form IFS theory that will be objects of discussion in the subsequent sections, which will be devoted to the software implementation of these basic concepts.

2.1 Definition of an IFS

Let a (crisp) set X be fixed and let A be a fixed symbol.

An IFS A^* in X is an object of the following form

$$A^* = \{\langle x, \mu_A(x), \nu_A(x)\rangle | x \in X\}, \tag{1}$$

where functions $\mu_A : X \to [0, 1]$ and $\nu_A : X \to [0, 1]$ define the *degree of membership* and the *degree of non-membership* of the element $x \in X$ to the IFS A^*, respectively, and for every $x \in X$

$$0 \leq \mu_A(x) + \nu_A(x) \leq 1. \tag{2}$$

Obviously, every ordinary fuzzy set has the form

$$\{\langle x, \mu_A(x), 1 - \mu_A(x)\rangle | x \in X\}.$$

If

$$\pi_A(x) = 1 - \mu_A(x) - \nu_A(x), \tag{3}$$

then $\pi_A(x)$ is the *degree of non-determinacy* (uncertainty) of the membership of element $x \in X$ to set A. In the case of ordinary fuzzy sets, $\pi_A(x) = 0$ for every $x \in X$.

Three very important notions to be used throughout the text are the following:

$$O^*_X(\text{ or } O^*(X)) := \{\langle x, 0, 1\rangle \mid x \in X\}, \tag{4}$$

$$E^*_X(\text{ or } E^*(X)) := \{\langle x, 1, 0\rangle \mid x \in X\}, \tag{5}$$

$$U^*_X(\text{ or } U^*(X)) := \{\langle x, 0, 0\rangle \mid x \in X\}, \tag{6}$$

Let us recall two of the main operators, introduced by Atanassov in [2], called modal operators. Necessity and possibility operators (denoted by $\Box$ and $\diamondsuit$, respectively) applied on an intuitionistic fuzzy set $A \in IFS(X)$ have been defined as:

$$\Box A = \{\langle x, \mu_A(x), 1 - \mu_A(x)\rangle | x \in X\}, \tag{7}$$

$$\diamondsuit A = \{\langle x, 1 - \nu(x), \nu_A(x)\rangle | x \in X\} \tag{8}$$

From the above definition it is evident that

$$\star \colon IFS(X) \longrightarrow FS(X),$$

where $\star$ is the prefix operator $\star \in \{\Box, \diamondsuit\}$, operating on the class of intuitionistic fuzzy sets.

2.2 Geometrical interpretations of an IFS

There are several geometrical interpretations of the IFSs (cf. Atanassov [9]), the earliest of which in the literature is the preprint from 1989 [3]. The three most relevant of them are discussed below (Figs. 1, 2, 4).

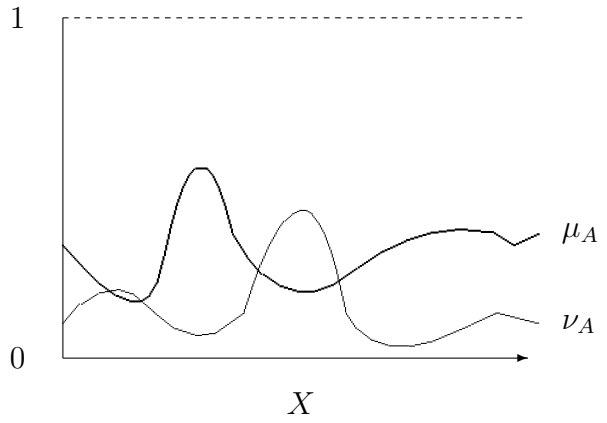


Figure 1. Standard geometrical representation of the membership degree μ_A and the non-membership degree ν_A.

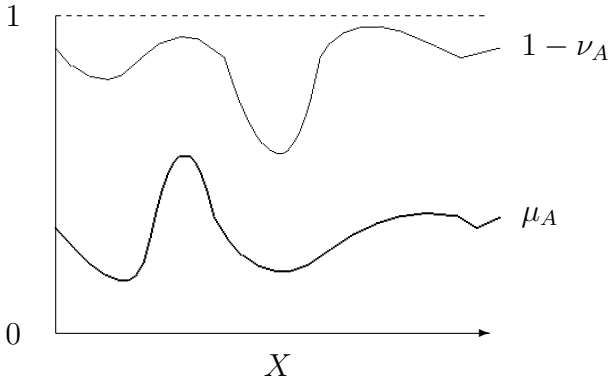
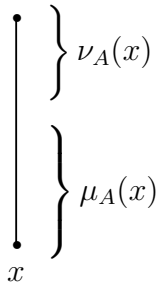


Figure 2. Modified geometrical representation of μ_A and $1 - \nu_A$.

Therefore, to every element $x \in X$ we can map a unit segment of the form:



On the other hand, the situation in Fig. 3 is impossible.

Let a universe X be given and let us consider the figure F in the Euclidean plane with a Cartesian coordinate system (see Fig. 4). Let us call the triangle F *"IFS-interpretational triangle"*. If $A \in IFS(X)$ is a fixed intuitionistic fuzzy set in the universe X, then we can construct a function $f_A : X \to F$ such that if $x \in X$, then

$$f_A(x) = \langle \mu_A(x), \nu_A(x) \rangle \in F. \tag{9}$$

And since $0 \leq \mu_A(x) + \nu_A(x) \leq 1$ (2), therefore the range of the function f_A is indeed a subset of F.

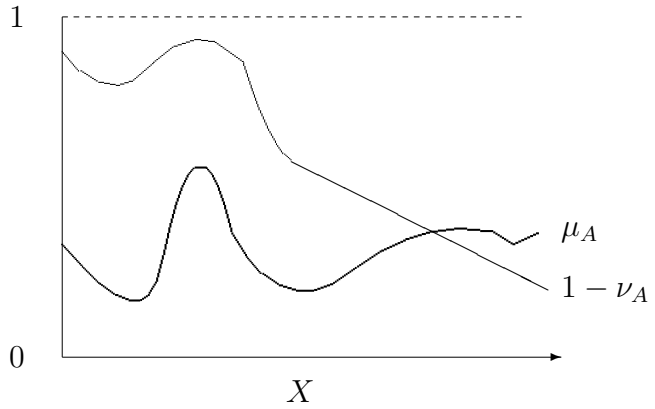


Figure 3. Representation of an impossible situation for μ_A and $1 - \nu_A$.

Note that if there exist two different elements $x_1, x_2 \in X$, $x_1 \neq x_2$, for which $\mu_A(x_1) = \mu_A(x_2)$ and $\nu_A(x_1) = \nu_A(x_2)$ with respect to some set $A \in IFS(X)$, then $f_A(x_1) = f_A(x_2)$.

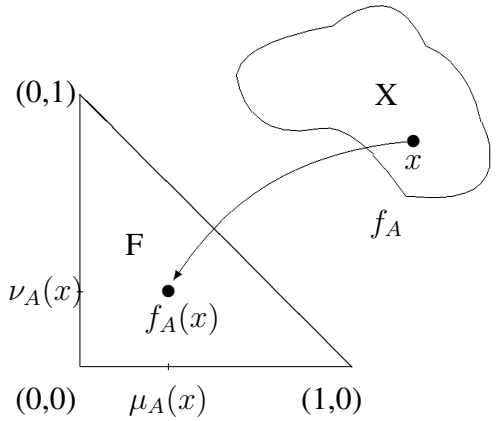


Figure 4. Triangular representation of a point from X in F.

2.3 Main operations on IFSs and their geometrical presentations

In this section, we give the most common operations on IFSs and their geometrical interpretations.

Following [2, 4, 9], for every two IFSs A and B the following relations and operations can be defined (everywhere below "iff" means "if and only if"). They are analogous of the standard set theoretical operations of *inclusion* (cf. [9]):

$$A \subseteq B \text{ iff } (\forall x \in X)(\mu_A(x) \leq \mu_B(x) \,\&\, \nu_A(x) \geq \nu_B(x)) \tag{10}$$

$$A \supseteq B \text{ iff } B \subseteq A \tag{11}$$

$$A = B \text{ iff } (\forall x \in X)(\mu_A(x) = \mu_B(x) \,\&\, \nu_A(x) = \nu_B(x)) \tag{12}$$

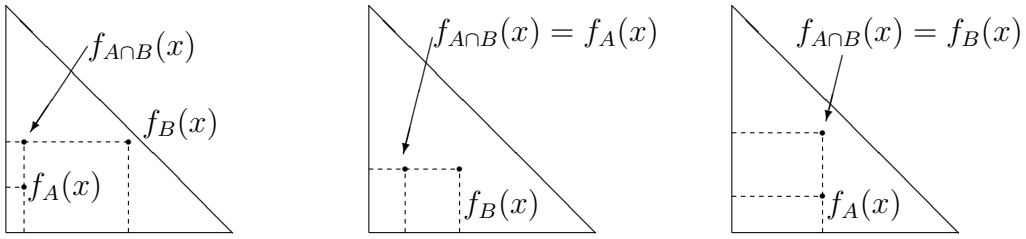
Let us now give the standard operations of *intersection*, *union*, as shown on Fig. 5 and Fig. 6 which are specific for the IFSs (cf. [9]):

$$A \cap B = \{\langle x, \min(\mu_A(x), \mu_B(x)), \max(\nu_A(x), \nu_B(x))\rangle | x \in X\} \tag{13}$$

$$A \cup B = \{\langle x, \max(\mu_A(x), \mu_B(x)), \min(\nu_A(x), \nu_B(x))\rangle | x \in X\} \tag{14}$$

If A and $B \in IFS(X)$, then a function $f_{A\cap B}$ assigns to $x \in X$, a point $f_{A\cap B}(x) \in F$ with coordinates

$$\langle \min(\mu_A(x), \mu_B(x)), \max(\nu_A(x), \nu_B(y))\rangle.$$

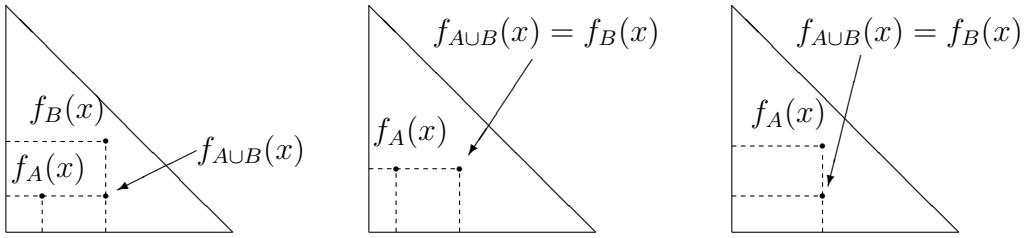


(a) (b) (c)

Figure 5. Representation of the intersection operation $\cap$ between A and $B \in IFS(X)$.

If A and B are two IFSs over X, then a function $f_{A\cup B}$ assigns to $x \in X$, a point $f_{A\cup B}(x) \in F$ with coordinates

$$\langle \max(\mu_A(x), \mu_B(x)), \min(\nu_A(x), \nu_B(y))\rangle$$



(a) (b) (c)

Figure 6. Representation of the union operation $\cup$ between A and $B \in IFS(X)$.

3 Standard topological operators on IFSs

Following [9], we introduce two topological operators. For every IFS A,

$$\mathcal{C}(A) = \{\langle x, K, L\rangle | x \in X\}, \tag{15}$$

where

$$K = \sup_{y \in X} \mu_A(y) \tag{16}$$

$$L = \inf_{y \in X} \nu_A(y) \tag{17}$$

and

$$\mathcal{I}(A) = \{\langle x, k, l\rangle | x \in X\}, \tag{18}$$

where

$$k = \inf_{y \in E} \mu_A(y), \tag{19}$$

$$l = \sup_{y \in E} \nu_A(y). \tag{20}$$

The following operators are defined in Atanassov [9], as extensions of the two topological operators C and I:

$$\mathcal{C}_\mu(A) = \{\langle x, K, \min(1 - K, \nu_A(x))\rangle | x \in X\}; \tag{21}$$

$$\mathcal{C}_\nu(A) = \{\langle x, \mu_A(x), L\rangle | x \in X\}; \tag{22}$$

$$\mathcal{I}_\mu(A) = \{\langle x, k, \nu_A(x)\rangle | x \in X\}; \tag{23}$$

$$\mathcal{I}_\nu(A) = \{\langle x, \min(1 - l, \mu_A(x)), l\rangle | x \in X\}, \tag{24}$$

where K, L, k, l have the forms (16), (17), (19), (20), respectively.

The geometrical interpretations of these operators applied on the IFS A in Fig. 7 are shown in Fig. 8 (a) and (b) and Fig. 9 (a) and (b).

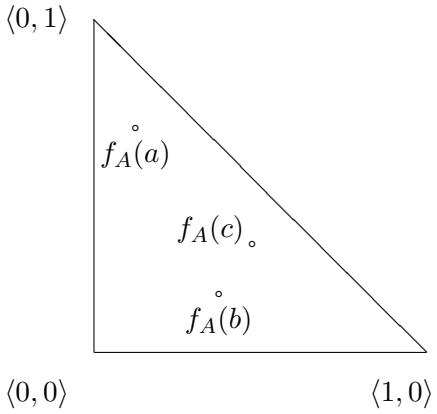


Figure 7. Example of an IFS on which topological operators will be applied.

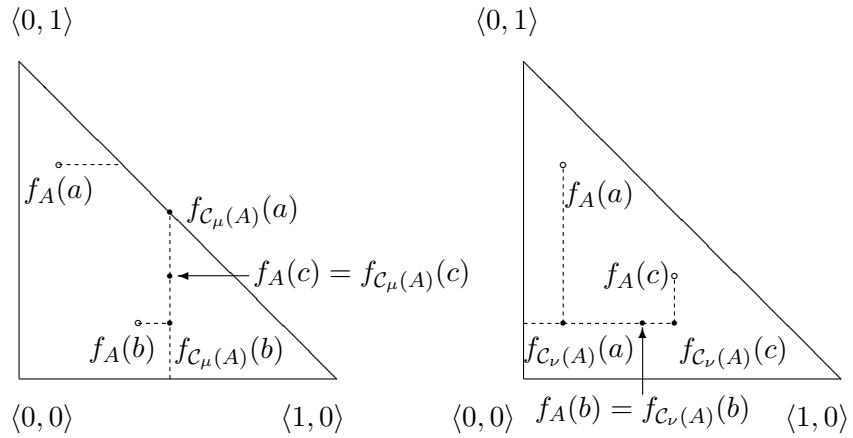


Figure 8. Extended closure operators $\mathcal{C}_\mu$ and $\mathcal{C}_\nu$.

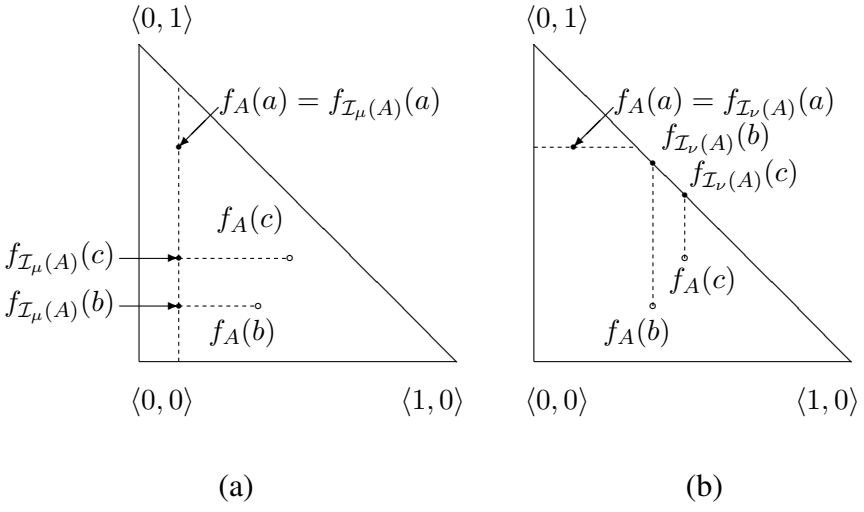


Figure 9. Extended interior operators $\mathcal{I}_\mu$ and $\mathcal{I}_\nu$.

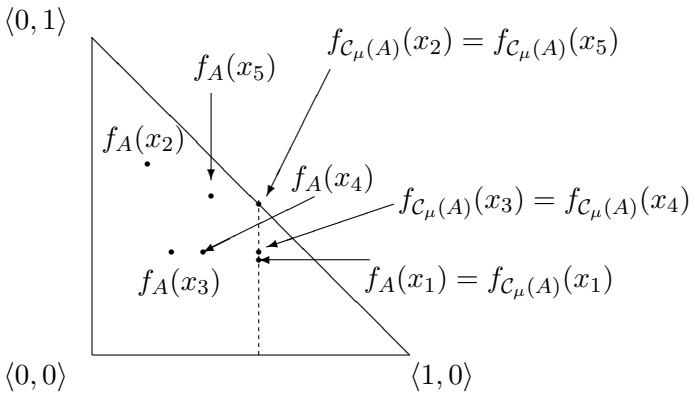


Figure 10. Example of $\mathcal{C}_\mu$.

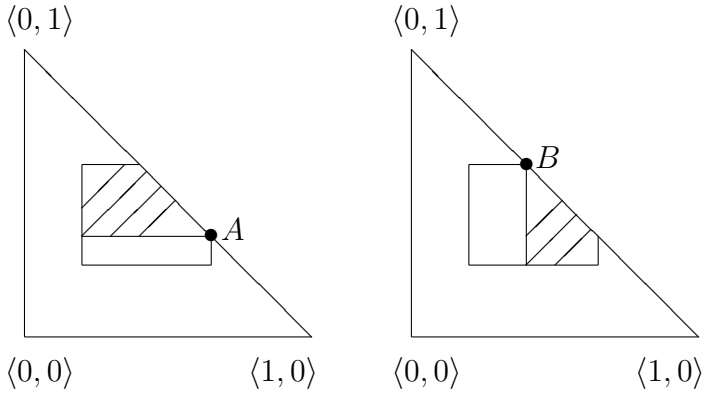
Another example of the action of operator $\mathcal{C}_\mu(A)$ is illustrated in Fig. 10. It is obvious from Fig. 10 that the operator $\mathcal{C}_\mu$ transforms points x_2 and x_5 to one point on the hypotenuse. More generally, all points from the hatching trapezoid can be transformed to point A in Fig. 11 (a). Similar is the situation in Fig. 11 (b), where all points from the hatching trapezoid can be transformed to point B by the operator $\mathcal{I}_\nu$.

In [8], the two new topological operators $\mathcal{C}^*_\mu$ and $\mathcal{I}^*_\nu$ are introduced as

$$\mathcal{C}^*_\mu(A) = \{\langle x, \min(K, 1-\nu_A(x)), \min(1-K, \nu_A(x))\rangle | x \in E\}; \tag{25}$$

$$\mathcal{I}^*_\nu(A) = \{\langle x, \min(1-l, \mu_A(x)), \min(l, 1-\mu_A(x))\rangle | x \in E\}, \tag{26}$$

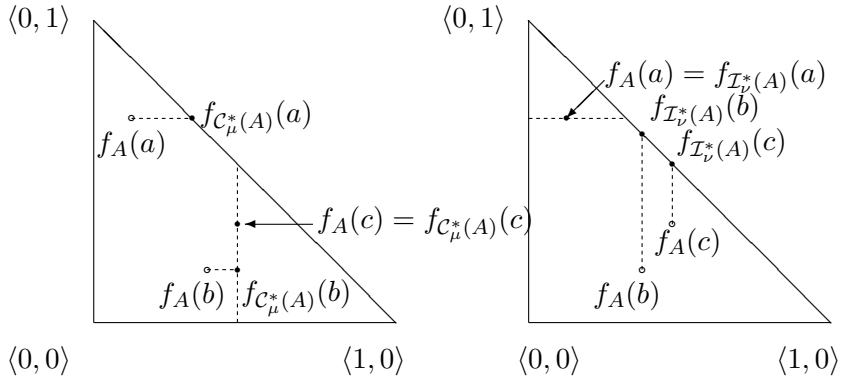
where K, L, k, l were already defined.



(a) (b)

Figure 11. High level representation of the operators $\mathcal{C}_\mu$ and $\mathcal{I}_\mu$.

The geometrical interpretations of the new operators applied on the IFS A having the form in Fig. 7 are given in Fig. 12 (a) and (b).



(a) (b)

Figure 12. Extended closure $\mathcal{C}^*_\mu$ and interior $\mathcal{I}^*_\nu$ operators applied on the IFS from Fig. 7.

In [10], Atanassov and Ban introduced the "weight-center operator" over a given IFS A by:

$$W(A) = \left\{ \left\langle x, \frac{\sum\limits_{y \in X} \mu_A(y)}{card(X)}, \frac{\sum\limits_{y \in X} \nu_A(y)}{card(X)} \right\rangle | x \in X \right\}, \tag{27}$$

where $card(X)$ is the number of the elements of a finite set X. For the continuous case, the "summation" may be replaced by integration over X.

3.1 Extended modal operators

Following Atanassov [6, 9], we construct a series of operators in the next subsections. Some of the extended modal operators will be defined in a more suitable way to be considered in the framework of the topological neighbourhoods and their properties. The change is a trivial linear transformation of the constants α and β, so that if $\langle \alpha, \beta \rangle = \langle 0, 0 \rangle$ the operator will be the identity.

3.1.1 Operators $\mathcal{D}_\alpha$ and $\mathcal{F}_{\alpha,\beta}$

The next operator represents both operators $\Box$ from (7) and $\diamondsuit$ from (8). Let $\alpha \in [0, 1]$ be a fixed real number. Given an IFS A, we define an operator $\mathcal{D}_\alpha$ as follows:

$$\mathcal{D}_\alpha(A) = \{\langle x, \mu_A(x) + \alpha.\pi_A(x), \nu_A(x) + (1 - \alpha).\pi_A(x) \rangle | x \in X\}. \tag{28}$$

From this definition it follows that $\mathcal{D}_\alpha(A)$ is a fuzzy set, because:

$$\mu_A(x) + \alpha.\pi_A(x) + \nu_A(x) + (1 - \alpha).\pi_A(x) = \mu_A(x) + \nu_A(x) + \pi_A(x) = 1.$$

To every point $x \in X$ the operator $f_{\mathcal{D}_\alpha(A)}$ assigns a point of the segment between $f_{\Box A}(x)$ and $f_{\diamondsuit A}(x)$ in the triangle F from Fig. 4 depending on the value of the argument $\alpha \in [0, 1]$ (see Fig. 13). As in the case of some of the above operations, this construction needs auxiliary elements which are shown in Fig. 13. As we noted above, the operator $\mathcal{D}_\alpha$ is an extension of the operators $\Box$ and $\diamondsuit$, but it can be extended even further.

Let $\alpha, \beta \in [0, 1]$ and $\alpha + \beta \leq 1$. Define (see [9]) the operator $\mathcal{F}_{\alpha,\beta}$, for the IFS A, by

$$\mathcal{F}_{\alpha,\beta}(A) = \{\langle x, \mu_A(x) + \alpha.\pi_A(x), \nu_A(x) + \beta.\pi_A(x) \rangle | x \in X\}. \tag{29}$$

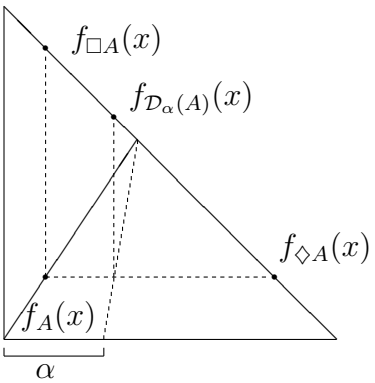


Figure 13. Extended modal operator $\mathcal{D}_\alpha(A)$.

To every point $x \in X$ the operator $f_{\mathcal{F}_{\alpha,\beta}(A)}$ assigns a point of the triangle with vertices $f_A(x)$, $f_{\Box A}(x)$ and $f_{\Diamond A}(x)$, depending on the value of the arguments $\alpha, \beta \in [0,1]$ for which $\alpha + \beta \leq 1$ (see Fig. 14).

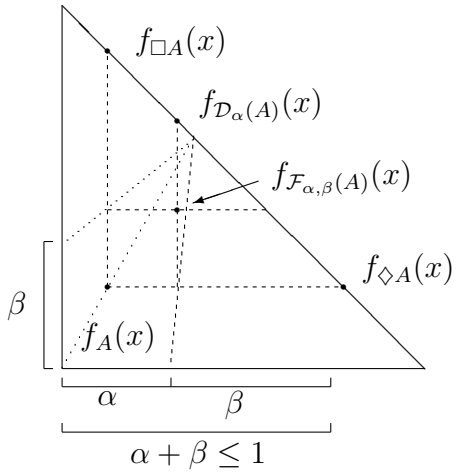


Figure 14. Extended modal operators $\mathcal{F}_{\alpha,\beta}$ and $\mathcal{D}_\alpha$

A feature that both operators share, together with the first two modal operators, is that each of them changes the degree of uncertainty. While the first three operators make this degree equal to zero, the operator $\mathcal{F}_{\alpha,\beta}$ only decreases its value, increasing the degrees of membership and non-membership of the IFS' elements.

3.1.2 Operator $\mathcal{G}_{\alpha,\beta}$

Let α, $\beta \in$ [0,1]. We are going to change the operator defined by Atanassov (cf. [9])

$$G_{\alpha,\beta}(A) = \{\langle x, \alpha\mu_A(x), \beta\nu_A(x)\rangle \,|\, x \in X\}$$

through the substitutions:

$$\alpha \to (1-\alpha) \text{ and } \beta \to (1-\beta)$$

That way the above extended modal operator will look like,

$$\mathcal{G}_{\alpha,\beta}(A) = \{\langle x, (1-\alpha)\mu_A(x), (1-\beta)\nu_A(x)\rangle \,|\, x \in X\} \tag{30}$$

We made this substitution in order to start at A for $(\alpha, \beta) = (0,0)$ and gradually to reach U_X^*. Obviously, $\mathcal{G}_{0,0}(A) = A$ and $\mathcal{G}_{1,1}(A) = U_X^*$, where U_X^* is defined by (6).

The operator f assigns a point $f_{\mathcal{G}_{\alpha,\beta}}(x)$ in the rectangle with vertex $f_A(x)$ and vertices with coordinates, $\langle pr_1 f_A(x), 0\rangle$, $\langle 0, pr_2 f_A(x)\rangle$ and $\langle 0,0\rangle$, where $pr_i p$ is the i-th projection $(i = 1,2)$ of the point p, to every point $x \in X$, depending on the value of the arguments $\alpha, \beta \in [0,1]$

Let $n \geq 1$ be an integer and $\alpha_i, \beta_i \in [0,1], i = 1, \ldots, n$. Then, we can construct the IFS

$$\mathcal{G}_{\alpha_n,\beta_n}(\ldots(\mathcal{G}_{\alpha_1,\beta_1}(A))\ldots)$$

$$= \{\langle x, \mu_A(x) \prod_{i=1}^{n} \alpha_i, \nu_A(x) \prod_{i=1}^{n} \beta_i \rangle | x \in E\} = \mathcal{G}_{\prod_{i=1}^{n} \alpha_i, \prod_{i=1}^{n} \beta_i}(A).$$

In [19], Vassilev studied some properties of this IFS.

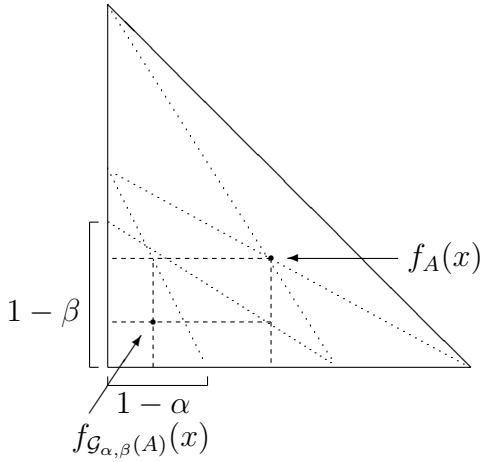


Figure 15. Extended modal operator $\mathcal{G}_{\alpha,\beta}$

3.1.3 Operators $\mathcal{H}_{\alpha,\beta}$, $\mathcal{H}^*_{\alpha,\beta}$, $\mathcal{J}_{\alpha,\beta}$, and $\mathcal{J}^*_{\alpha,\beta}$

We state here four other extended modal operators over an IFS A, given the fixed real numbers α, $\beta \in$ [0,1] (see Atanassov [9]), as

$$H_{\alpha,\beta}(A) = \{\langle x, \alpha\mu_A(x), \nu_A(x) + \beta\pi_A(x)\rangle | \, x \in X\};$$
$$H^*_{\alpha,\beta}(A) = \{\langle x, \alpha\mu_A(x), \nu_A(x) + \beta(1 - \alpha\mu_A(x) - \nu_A(x))\rangle \, | \, x \in X\};$$
$$J_{\alpha,\beta}(A) = \{\langle x, \mu_A(x) + \alpha\pi_A(x), \beta\nu_A(x)\rangle \, | \, x \in X\};$$
$$J^*_{\alpha,\beta}(A) = \{\langle x, \mu_A(x) + \alpha(1 - \mu_A(x) - \beta\nu_A(x)), \beta\nu_A(x)\rangle \, | \, x \in X\};$$

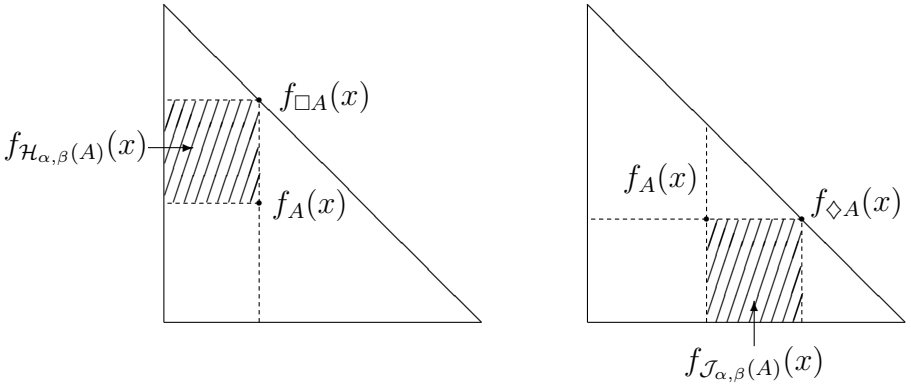
In the above definitions for $H_{\alpha,\beta}$ and $H^*_{\alpha,\beta}$ we make the substitution: $\alpha \to (1 - \alpha)$ and for $J_{\alpha,\beta}$ and $J^*_{\alpha,\beta}$ we make the substitution: $\beta \to (1 - \beta)$. That way we get the last four extended modal operators:

$$\mathcal{H}_{\alpha,\beta}(A) = \{\langle x, (1 - \alpha)\mu_A(x), \nu_A(x) + \beta\pi_A(x)\rangle | \, x \in X\};$$
$$\mathcal{H}^*_{\alpha,\beta}(A) = \{\langle x, (1 - \alpha)\mu_A(x), \nu_A(x) + \beta(1 - (1 - \alpha)\mu_A(x) - \nu_A(x))\rangle \, | \, x \in X\};$$
$$\mathcal{J}_{\alpha,\beta}(A) = \{\langle x, \mu_A(x) + \alpha\pi_A(x), (1 - \beta)\nu_A(x)\rangle \, | \, x \in X\};$$
$$\mathcal{J}^*_{\alpha,\beta}(A) = \{\langle x, \mu_A(x) + \alpha(1 - \mu_A(x) - (1 - \beta)\nu_A(x)), (1 - \beta)\nu_A(x)\rangle \, | \, x \in X\};$$

We made those substitutions because we want to start the mapping from $\langle \mu_A, \nu_A \rangle$ at $\langle \alpha, \beta \rangle = \langle 0, 0 \rangle$.

The operator $f_{\mathcal{H}_{\alpha,\beta}(A)}$ assigns to every point $x \in X$ a point $f_{\mathcal{H}_{\alpha,\beta}(A)}(x)$ of the rectangle with vertices with coordinates $\langle 0, pr_2 f_A(x)\rangle$, $\langle 0, pr_2 f_{\Box A}(x)\rangle$ and vertices $f_{\Box A}(x)$ and $f_A(x)$, depending on the value of the parameters $\alpha, \beta \in [0, 1]$ (see Fig. 16 (a)).

The operator $f_{\mathcal{J}_{\alpha,\beta}(A)}$ assigns to every point $x \in X$ a point $f_{\mathcal{J}_{\alpha,\beta}(A)}(x)$ of the rectangle with vertices with coordinates $\langle pr_1 f_{\diamondsuit A}(x), 0\rangle, \langle pr_1 f_A(x), 0\rangle$ and vertices $f_A(x)$ and $f_{\diamondsuit A}(x)$, depending on the value of the parameters $\alpha, \beta \in [0,1]$ (see Fig. 16 (b)).

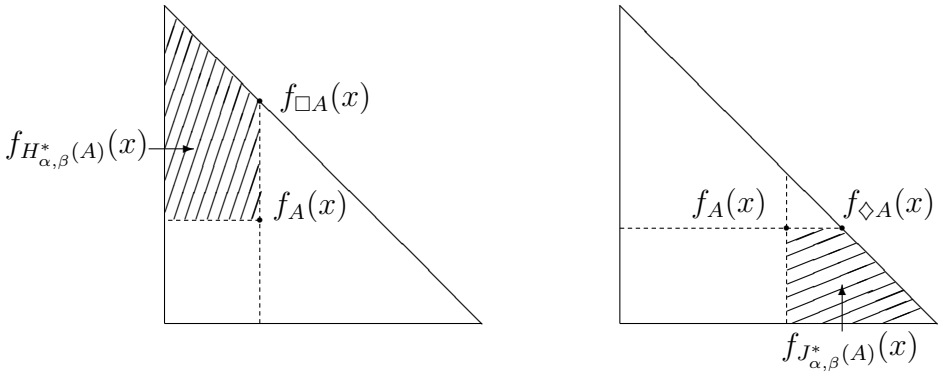


(a) (b)

Figure 16. Extended operators $\mathcal{H}_{\alpha,\beta}$ and $\mathcal{H}_{\alpha,\beta}$

The operator $f_{\mathcal{H}^*_{\alpha,\beta}(A)}$ assigns to every point $x \in X$ a point $f_{\mathcal{H}^*_{\alpha,\beta}(A)}(x)$ from the figure with vertices with coordinates $\langle 0, pr_2 f_A(x)\rangle$ and $\langle 0, 1\rangle$ and vertices $f_{\Box A}(x)$ and $f_A(x)$, depending on the value of the parameters $\alpha, \beta \in [0,1]$ (see Fig. 17 (a)).

The operator $f_{\mathcal{J}^*_{\alpha,\beta}(A)}$ assigns to every point $x \in E$ a point $f_{\mathcal{J}^*_{\alpha,\beta}(A)}(x)$ from the figure with vertices with coordinates $\langle 1, 0\rangle$ and $\langle pr_1 f_A(x), 0\rangle$ and vertices $f_A(x)$ and $f_{\diamondsuit A}(x)$, depending on the value of the parameters $\alpha, \beta \in [0,1]$ (see Fig. 17 (b)).



(a) (b)

Figure 17. Extended operators $\mathcal{H}^*_{\alpha,\beta}$ (a) and $\mathcal{J}^*_{\alpha,\beta}$ (b)

In this paper we have stated the definition, theory, applications and main properties of IFSs, as introduced by Atanassov in 1983 [2,4,9]. Many of the properties and operators presented here are used throughout the presented software implementation. And the (pre)topological operators introduced in [16] are related to the extended topological and extended modal (cf. [5,7]) operators. The pretopological (preinterior and preclosure) operators are defined as follows:

For $\alpha, \beta, \gamma_\alpha, \gamma_\beta \in [0,1]$ we define the preinterior operator

$$\mathcal{I}^{\gamma_\alpha,\gamma_\beta}_{\mu;\alpha,\beta} : IFS(X) \longrightarrow IFS(X),$$

such that

$$\mu_{\mathcal{I}^{\gamma_\alpha,\gamma_\beta}_{\mu;\alpha,\beta}(A)}(x) = \begin{cases} 0 & \text{if } 0 \leq \mu_A(x) < \gamma_\alpha.\alpha \\ \frac{1}{1-\gamma_\alpha}(\mu_A(x) - \alpha) + \alpha & \text{if } \gamma_\alpha.\alpha \leq \mu_A(x) < \alpha \\ \mu_A(x) & \text{if } \alpha \leq \mu_A(x) \leq 1 \end{cases} \tag{31}$$

$$\nu_{\mathcal{I}^{\gamma_\alpha,\gamma_\beta}_{\mu;\alpha,\beta}(A)}(x) = \begin{cases} \min((1-\gamma_\beta)\nu_A(x) + \beta\gamma_\beta, 1 - \mu_{\mathcal{I}^{\gamma_\alpha,\gamma_\beta}_{\mu;\alpha,\beta}(A)}(x)) & \text{if } 0 \leq \nu_A(x) \leq \beta \\ \nu_A(x) & \text{if } \beta < \nu_A(x) \leq 1 \end{cases} \tag{32}$$

For $\alpha, \beta, \gamma_\alpha, \gamma_\beta \in [0,1]$ we define the preclosure operator

$$\mathcal{C}^{\gamma_\alpha,\gamma_\beta}_{\nu;\alpha,\beta} : IFS(X) \longrightarrow IFS(X),$$

such that

$$\nu_{\mathcal{C}^{\gamma_\alpha,\gamma_\beta}_{\nu;\alpha,\beta}(A)}(x) = \begin{cases} 0 & \text{if } 0 \leq \nu_A(x) < \gamma_\alpha.\alpha \\ \frac{1}{1-\gamma_\alpha}(\nu_A(x) - \alpha) + \alpha & \text{if } \gamma_\alpha.\alpha \leq \nu_A(x) < \alpha \\ \nu_A(x) & \text{if } \alpha \leq \nu_A(x) \leq 1 \end{cases} \tag{33}$$

$$\mu_{\mathcal{C}^{\gamma_\alpha,\gamma_\beta}_{\nu;\alpha,\beta}(A)}(x) = \begin{cases} \min((1-\gamma_\beta)\mu_A(x) + \beta\gamma_\beta, 1 - \nu_{\mathcal{C}^{\gamma_\alpha,\gamma_\beta}_{\nu;\alpha,\beta}(A)}(x)) & \text{if } 0 \leq \mu_A(x) \leq \beta \\ \mu_A(x) & \text{if } \beta < \mu_A(x) \leq 1 \end{cases} \tag{34}$$

4 Software implementation of the operators for IFS

In this section, we explain how the software implementation can be used. It consists of open source codes that can be freely used and further customized by the users. The IFSs and their operators have been modelled in the programming language Python and visualized through the library `matplotlib`.

We are going to briefly describe the most important python scripts and definition of object and their main functionalities.

4.1 Universal set

The script **universal_set.py** consists of the definition of the main object describing an IFS and main operations for their mutation and consistency checks.

- Add new object to the Universe
- Saving the universe in CSV format
- Set universe to the object
- Get the lenght of Universe
- Check if the Universe is empty

```
####  @@@@@@@@@@   @@@@@@@@@@@@@    @@@@@@@@@@@@@@@
####   @@@@@@@@@@ universal_set.py @@@@@@@@@@@@@@@
####  @@@@@@@@@@   @@@@@@@@@@@@@    @@@@@@@@@@@@@@@

class UniversalSet(object)
  (words_set=None):
  def __eq__(self, other):
  def __len__(self):
  def __getitem__(self, idx):
  def to_csv(self, file_name, index=True):
  def indices(self):
  def indices_generator(self):
  def add(self, words_set, return_flag=None):
  def _add(self, words):
  def get_index(self, word, default=None):
  def get_word(self, idx, default=None):
  def characteristic_indices(self, words):
  def check_consistancy(self):
  def length(self):
  def empty(self):
  def set_universe(self,
    index_to_words=[],
    words_to_index={},
    rhs=None,
    copy_op=lambda x: x):
  """Sets the universal set through the corresponding index_to_words
  (The words of the universe have to be ordered and therefore they
  posses an index in this ordering) or words_to_index.

  Attributes
  ----------
  index_to_words : list with the words, occuring by their corresponding order
  words_to_index : dictionary with the words as keys and their corresponding
  values
  copy_op : Operation that determines how the universal set to be stored.
  It can be stored as a hard/deep/ copy into the member variables of self
  /the current class/ or the corresponding member variable can point to the
  universal set from outside.
  """
```

In the next script the LATTICE structure of the IFSs is taken into account and two main orderings between IFSs are defined as posets (partially ordered sets). The main operations for posets: the standard one and the π-ordering as described in [15].

- Infimum and Supremum of a subset of the Universe
- equality
- less
- less or equal
- greater
- greater or equal

```
####  @@@@@@@@@@   @@@@@@@@@@@@    @@@@@@@@@@@@@@@
####   @@@@@@@@@@   lattice.py    @@@@@@@@@@@@@@@
####  @@@@@@@@@@   @@@@@@@@@@@@    @@@@@@@@@@@@@@@

# Triangular Poset - partially ordered set
# CLASS
class TriangPoset(object):
__metaclass__  = abc.ABCMeta

  @abc.abstractmethod
  def eq(self, first, second):
  """
  Return
  True - iff the elements are equal in the poset.
  None - iff the elements are not comparable
  Raise exception iff they are not of comparable classes
  """
  return

  @abc.abstractmethod
  def leq(self, first, second):

  @abc.abstractmethod
  def sup(self, *args):

  @abc.abstractmethod
  def inf(self, *args):
  def neq(self, first, second):
  def lt(self, first, second):
  def geq(self, first, second):
  def gt(self, first, second):
  def is_correct(self, mu, nu):

# Standard Triangular Poset
# CLASS
class StdTriangPoset(TriangPoset):

  def eq(self, a, b):
  def leq(self, a, b):
  def sup(self, *args):
  def inf(self, *args):

# Pi Triangular Poset (Pi-Ordering)
# CLASS
class PiTriangPoset(TriangPoset):

  def eq(self, a, b):
  def leq(self, a, b):
  def sup(self, *args):
  def inf(self, *args):
```

4.2 IF stack bars representations and stack bars memberhsip and non-membership histogram

In the next script we introduce functions that plot two types of stack bars for IFSs. We also show a 2D histogram taking into account the membership and non-memberhsip functions.

Next visualizations can be reproduced by the command:

python3 ifs_test.py

- Type 1 visualization of an IFS, as shown on Fig. 18
- Type 2 visualization of an IFS, as shown on Fig. 19
- 2D Histogram of membeship and non-membership degrees, as shown on Fig. 20.

```
####  @@@@@@@@@@   @@@@@@@@@@@@    @@@@@@@@@@@@@@@
####   @@@@@@@@@@  ifs_2Dplot.py  @@@@@@@@@@@@@@@
####  @@@@@@@@@@   @@@@@@@@@@@@    @@@@@@@@@@@@@@@

def rotate_axislabels(ax, angles={'x': 45,'y': 45,'z': 45}):
def plot_grid_triangular(ax, rang, muEdges=None, nuEdges=None,
def plot_bar_type_2(ifs):
"""
plot stack bars type = 'interval' type only
"""
def plot_bar_type_1(ifs, plot_pi=False):
"""
plot stack bars type = 'intuitionistic' type only
"""
```

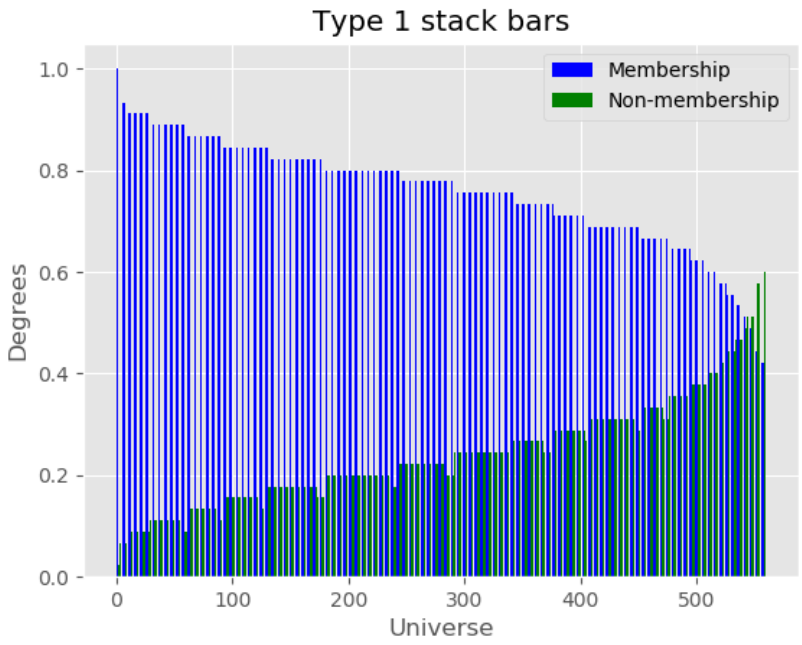


Figure 18. Plot of an IFS of type 1 stack bars.

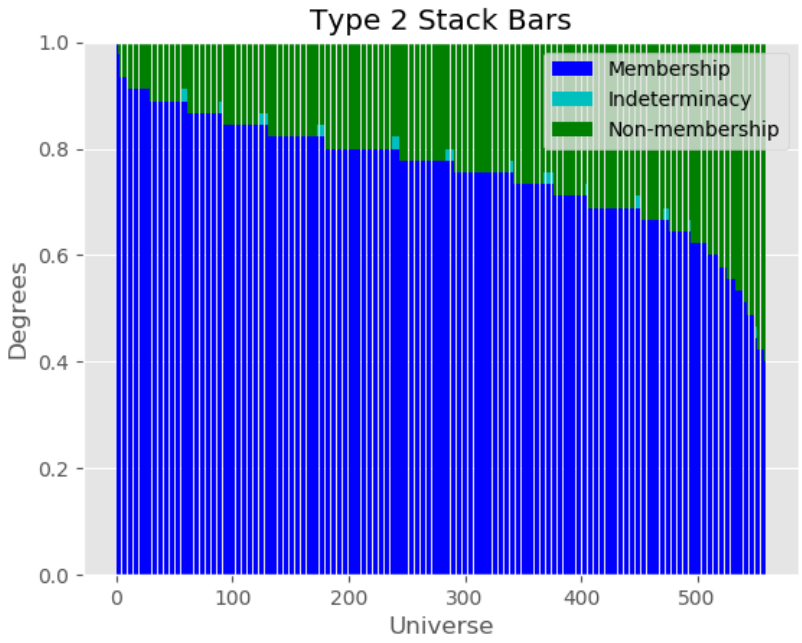


Figure 19. Plot of an IFS of type 2 stack bars.

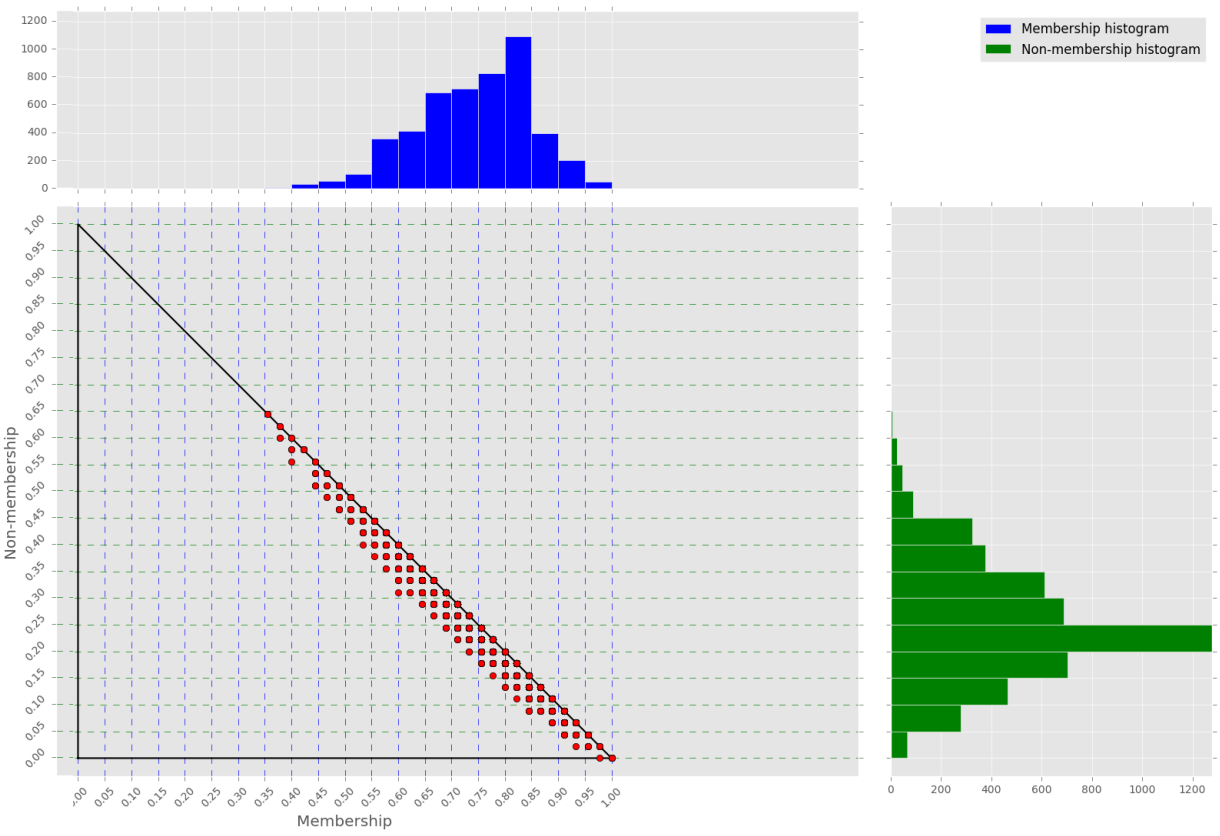


Figure 20. 2D Histogram: representing the histogram of membeship and non-membership degrees (see Fig. 21).

4.3 3D Histograms

In the next file are defined two types of 3D Histograms.

- The first one on Fig. 21 represents 3D Histogram for the membership and non-membership degrees.
- The second one on Fig. 22 represents 3D Histogram for 2D coordinates of the elements of the IFS represented in the triangle.

It is noteworthy that such 3D histograms for the degrees of intuitionistic fuzziness are of particular interest for the visualization of the results of intercriteria analysis as discussed in [11, 12] and further implemented in [18]. The novelty with the herewith proposed histogramic approach is the better visualization of the cases when otherwise distinct pairs of criteria form numerically identical intercriteria pairs.

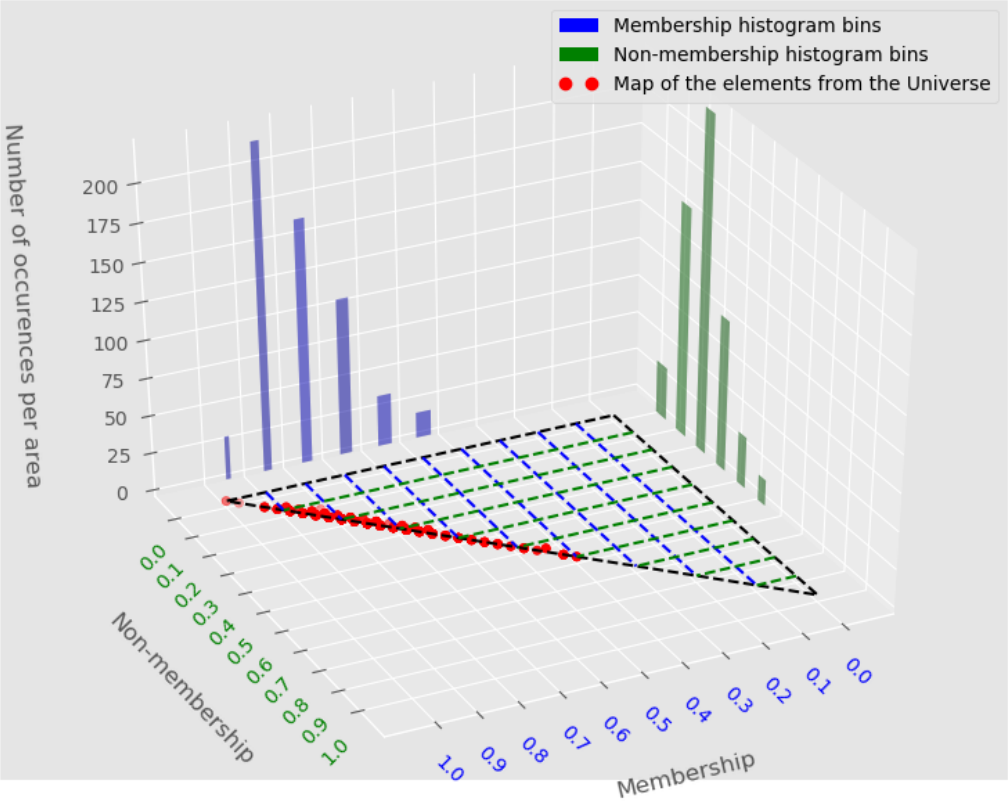


Figure 21. 3D plot of 2D histogram: representing the histogram of membeship and non-membership degrees (see Fig. 20)

```
####  @@@@@@@@@@   @@@@@@@@@@@@    @@@@@@@@@@@@@@@@
####   @@@@@@@@@@  ifs_3Dplot.py  @@@@@@@@@@@@@@@@
####  @@@@@@@@@@   @@@@@@@@@@@@    @@@@@@@@@@@@@@@@

def rotate_axislabels(ax, angles={'x': 45,'y': 45,'z': 45}):
def plot_grid_triangular(ax, rang, muEdges=None, nuEdges=None,
def plot_membership_3Dhistogram(ifs,
```

```
  def plot_membership(typ):

def plot_3D_histogramm(
  ifs,
  bins=None,
  colors={'mu':'b','nu':'g','hist':'y','elem':'r'}):
  '''
  Plot a 3D histogram in the triangular representation of an IFS.
  '''
```

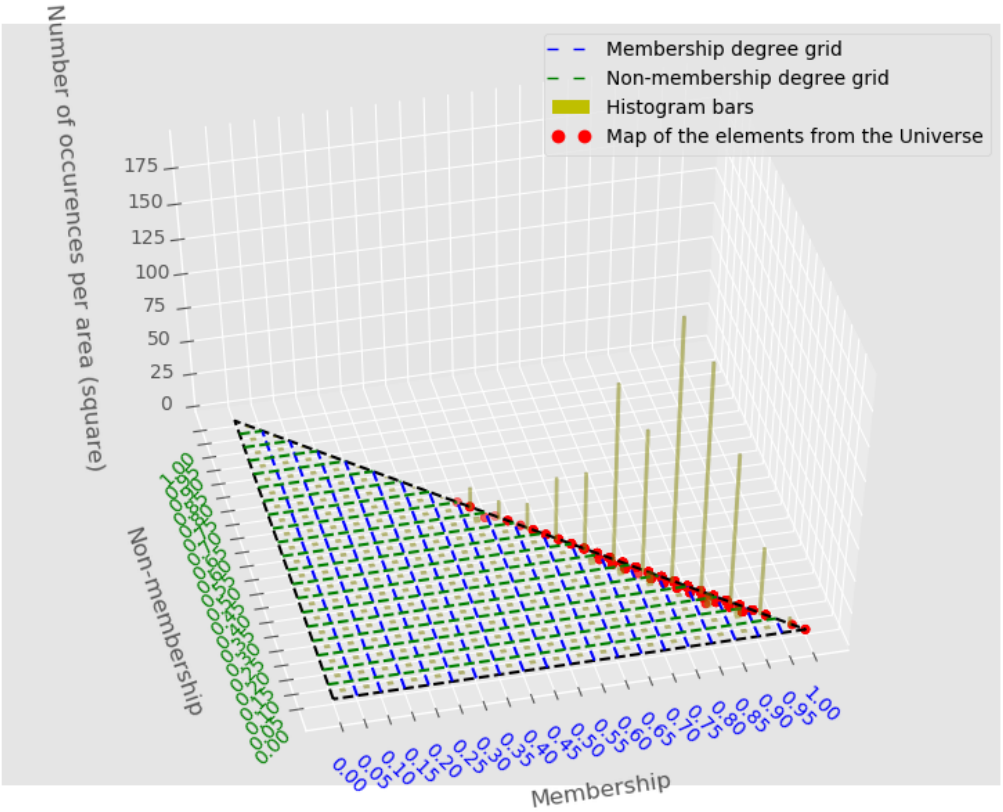


Figure 22. 3D Histogram.

4.4 Topological Operators tool

In this section we present the topological operators tool. Next visualizations can be reproduced by the command:

python3 ifs_topo_modeler.py

As we can see on Fig. 23, it is an interactive plot where one can change:

- font size of the corresponding point
- contrast of the corresponding point
- radius size of the corresponding point
- show / hide IFS

- show / hide labels
- save IFSs
- show / hide Closure and Interior operators

The user can interactively drag and drop the dark blue points in the triangle corresponding to the original IFSs. By pressing the button 'Save IFS' all the results are are saved in .json formats. That is, in the folder **.\working_dir** we can see the following files:

- **ifs_0000_original_.json**
- **ifs_0000_inc2_.json**
- **ifs_0000_cl2_.json**

The files representing the corresponding IFSs look like this:

```
ifs_0000_original_.json

{
  "data": {
    "0": [
    0.21579643297921236,
    0.13886340760817428
    ],
    "1": [
    0.00011437481734488664,
    0.6976674273681602
    ],
    "2": [
    0.5100635481451006,
    0.34367331976363236
    ],
    "3": [
    0.22285884374319376,
    0.378985373583539
    ],
    "4": [
    0.5571462865716426,
    0.12944685992286575
    ]
  },
  "contrast": 1,
  "labels_size": 12,
  "color": "blue",
  "marker_size": 0.01,
  "label": "ifs_0000_original"
}

ifs_0000_inc2_.json

{
```

```
  "data": {
    "0": [
    0.1368520471131605,
    0.13886340760817428
    ],
    "1": [
    0.00011437481734488664,
    0.6976674273681602
    ],
    "2": [
    0.5100635481451006,
    0.407469327905453
    ],
    "3": [
    0.1469412053474196,
    0.42159414943341567
    ],
    "4": [
    0.5571462865716426,
    0.12944685992286575
    ]
  },
  "contrast": 0.5,
  "labels_size": 12,
  "color": "magenta",
  "marker_size": 0.01,
  "label": "ifs_0000_inc2"
}

ifs_0000_cl2_.json

{
  "data": {
    "0": [
    0.2710575030854486,
    0.13886340760817428
    ],
    "1": [
    0.00011437481734488664,
    0.6976674273681602
    ],
    "2": [
    0.5100635481451006,
    0.25
    ],
    "3": [
    0.27600119062023565,
    0.27246343395884753
    ],
    "4": [
```

```
    0.5571462865716426,
    0.12944685992286575
    ]
  },
  "contrast": 0.8,
  "labels_size": 12,
  "color": "orange",
  "marker_size": 0.01,
  "label": "ifs_0000_cl2"
}
```

And the source code and images blow correspond to:

- A screenshot of the UI showing the result of the preinterior and preclosure operators on the IFS A with binary inclusion indicator metrics (as defined in [17]) of the IFS and its mappings are visualized (cf. [16]) as shown on Fig. 24.
- The raw functionality of the topological parameters and an IFS are as shown on Fig. 25.

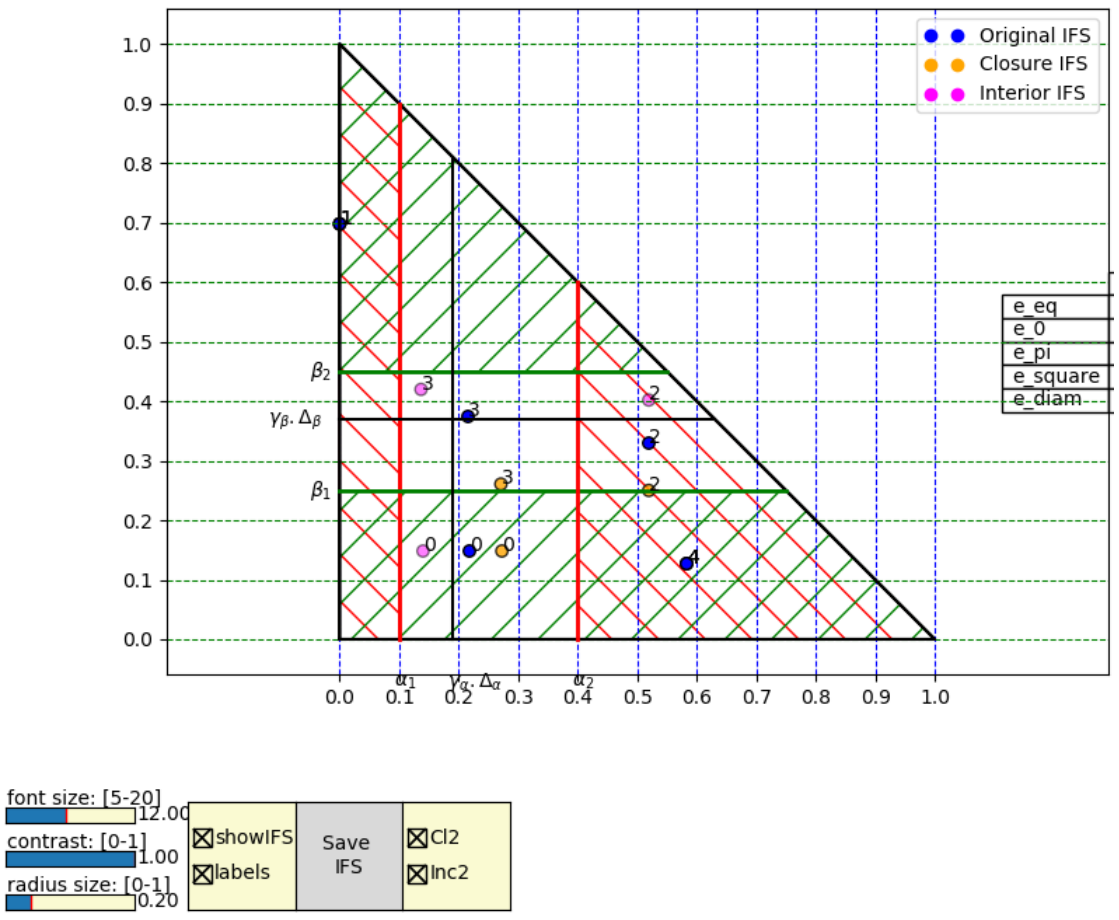
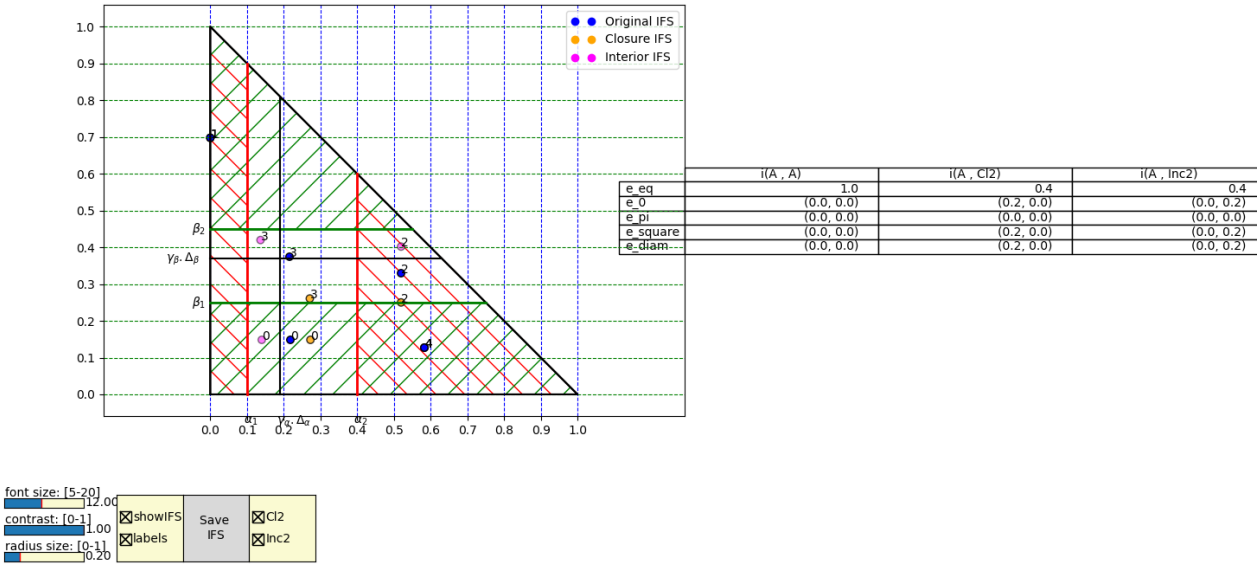


Figure 23. The results of preinterior operator $\mathcal{I}^{\gamma_\alpha,\gamma_\beta}_{\mu;\alpha,\beta}$ on $A \in IFS(X)$ and preclosure operator $\mathcal{C}^{\gamma_\alpha,\gamma_\beta}_{\nu;\alpha,\beta}$ on A (see [16]).



	i(A , A)	i(A , Cl2)	i(A , Inc2)
e_eq	1.0	0.4	0.4
e_0	(0.0, 0.0)	(0.2, 0.0)	(0.0, 0.2)
e_pi	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)
e_square	(0.0, 0.0)	(0.2, 0.0)	(0.0, 0.2)
e_diam	(0.0, 0.0)	(0.2, 0.0)	(0.0, 0.2)

Figure 24. The result of the preinterior and preclosure operators on the IFS A with binary inclusion indicator metrics (cf. [17]) of the IFS and its mappings visualized (cf. [16])

```
####  @@@@@@@@@@    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@     @@@@@@@@@@@@@@@@
####   @@@@@@@@@@  ifs_triangular_representation.py    @@@@@@@@@@@@@@@@
####  @@@@@@@@@@    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@     @@@@@@@@@@@@@@@@
```

```
class IfsTriangAbstract(object):
__metaclass__  = abc.ABCMeta

  @abc.abstractmethod
  def get_data(self):

  @abc.abstractmethod
  def get_data_pair(self):

class IfsTriang(IfsTriangAbstract):
flip = 1

  def __init__(self, axes, musnus,
    radius=0.01,
    label_id='ifs_001',
    labels=None,
    picker=10,
    alpha_marker=0.5,
    visible=True,
    annotation_size=12,
    show_annotation=True,
    colors = {'mu':'b', 'nu':'g', 'elem':'r'},
    bins = {'mu':10, 'nu':10},
    init_flag=True):

  self.holder.append(obj)
```

```
31
32    def get_color(self):
33    def get_visible(self):
34    def set_visible(self, visible):
35    def get_annotation_visible(self):
36    def set_annotation_visible(self, show_annotation):
37
38    def get_radius(self):
39    def set_radius(self, radius):
40    def get_annotation_size(self):
41    def set_annotation_size(self, annotation_size):
42
43    def get_alpha_marker(self):
44    def set_alpha_marker(self, alpha_contrast):
45    def get_data(self):
46    def get_data_pair(self):
47    def save_to_json(self, json_path):
48
49    @property
50    def default_path(self):
51    def save_to_json_default(self, event):
52
53 ##############################################################################
54
55 class IfsTriangInteractive(IfsTriang):
56    musnus,
57    radius=0.01,
58    companions=col.OrderedDict([]),
59    metrics_unary=col.OrderedDict([]),
60    metrics_binary=col.OrderedDict([]),
61    widgets=None,
62    label_id = '',
63    labels=None,
64    picker=10,
65    alpha_marker=0.5,
66    visible=True,
67    annotation_size=12,
68    show_annotation=True,
69    colors = {'mu':'b', 'nu':'g', 'elem':'r'},
70    bins = {'mu':10, 'nu':10},
71    init_flag=True
72    ):
73
74    # Connect to all the events we need
75    def connect(self):
76    def on_press(self, event):
77    def on_pick(self, event):
78
79    # on motion we will move the rect if the mouse is over the object
80    def on_motion(self, event):
```

```
  def sync_companions(self):
  def draw_blit(self, obj):
  def update_tables(self):

  # on release we reset the press data
  def on_release(self, event):

  def save_to_json_default(self, event):

################################################################################

class IfsTriangTopoConstInteractive(IfsTriangInteractive):
    def __init__(self,
    axes, axes_metrics,
    musnus,
    topo_const_triang,      # NEW related to TOPOCONST
    radius=0.01,
    companions=col.OrderedDict([]),
    metrics_unary=col.OrderedDict([]),
    metrics_binary=col.OrderedDict([]),
    widgets=None,
    label_id = '',
    labels=None,
    picker=10,
    alpha_marker=0.5,
    visible=True,
    annotation_size=12,
    show_annotation=True,
    colors = {'mu':'b', 'nu':'g', 'elem':'r'},
    bins = {'mu':10, 'nu':10},
    init_flag=True
    ):

  def connect(self):
    super(IfsTriangTopoConstInteractive, self).connect()
    self.topo_const_triang.connect()
```

```
####  @@@@@@@@@@  @@@@@@@@@@@@@@@@@@@@@@  @@@@@@@@@@@@@@@
####   @@@@@@@@@@ ifs_properties_topo.py @@@@@@@@@@@@@@@
####  @@@@@@@@@@  @@@@@@@@@@@@@@@@@@@@@@  @@@@@@@@@@@@@@@

@Define the ifs topological properties and the plots in the triangle
ifs_properties_topo.py

class TopoConst(object):
  (self, ax,
       alpha,
       beta,
       gamma_a,
       gamma_b,
```

```
14        companion=None,
15        label=None):
16
17    @classmethod
18    # CLASS CONSTRUCTOR
19    def from_json(cls, json_path, ax):
20
21    def save_to_json(self, json_path):
22
23    @property
24    def default_path(self):
25    def save_to_json_default(self, event):
26    def set_topoconst(self, alpha, beta):
27
28    def _fill_basic(self):
29    def fill_fixed(self, typ):
30    def fill_nu_full(self):
31    def fill_nu_limited(self):
32    def fill_mu_limited(self):
33    def fill_mu_line(self):
34    def fill_nu_line(self):
35    def fill_inclusion_indicator_interior(self):
36    def draw_topo_object(self):
37    def set_visible(self, flag):
38    def get_visible(self):
39
40 ##########################################################################
41
42 class TopoConstGeneral(object):
43    json_path_topoconst,
44    ax,
45    bins,
46    rotation,
47    color='b',
48    marker='o'):
49
50    def set_animated(self,value):
51    def set_annotations_general(self):
52
53    @classmethod
54    def from_json(cls, json_path, ax):
55    def save_to_json(self, json_path):
56    def save_to_json_default(self, event):
57    def set_topoconst(self, alpha, beta, alpha0, beta0):
58    def draw_topo_object(self):
59    def set_visible(self, flag):
60    def get_visible(self):
61
62 ##########################################################################
63
```

```
class TopoConstInteractive(TopoConst):
    (self, ax,
    alpha, beta,
    gamma_a, gamma_b,
    companion=None,
    label=None):

  # 'connect to all the events we need'
  def connect(self):
  def on_pick(self, event):

  # 'on motion we will move the rect if the mouse is over us'
  def on_motion(self, event):

  # 'on release we reset the press data'
  def on_release(self, event):
```

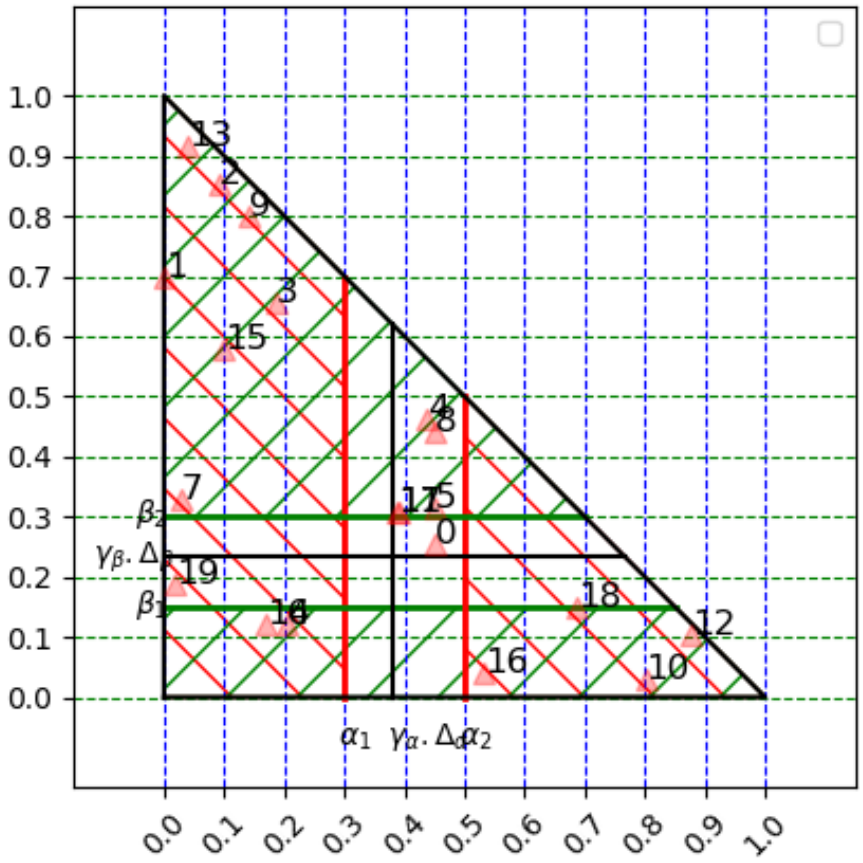


Figure 25. Raw functionality of the topological parameters and an IFS

```
####  @@@@@@@@@@  @@@@@@@@@@@@@@@@@@@@@@  @@@@@@@@@@@@@@@@
####   @@@@@@@@@@  ifs_properties_plot.py @@@@@@@@@@@@@@@@
####  @@@@@@@@@@   @@@@@@@@@@@@@@@@@@@@@@  @@@@@@@@@@@@@@@@

@ Define the widgets for the interactive plot
ifs_properties_plot.py
```

```
7
8  # Gives the main methods structure of the IFS proerty class
9  # CLASS
10 class PropertiesBasic
11
12   # ABSTRACT CLASS METHODS
13
14   @abc.abstractmethod
15   def get_data(self):
16
17   @abc.abstractmethod
18   def set_data(self, data):
19
20   @abc.abstractclassmethod
21   def get_color(self):
22
23   @abc.abstractclassmethod
24   def set_color(self):
25
26 # Extends the PropertiesBasic with annotations
27 # CLASS
28 class PropertiesAnnotations(PropertiesBasic)(
29   # Label of the IFS property
30   label=None,
31
32   # Holder of the IFS
33   holder=None,
34
35   # Radius of the 'points'
36   radius=5,
37
38   # Annotations (mainly numbers)
39   annotations=None,
40
41   # Brightness of the 'point'
42   alpha_marker=0.5,
43
44   # Size of the label
45   labels_size=12,
46
47   # Hide / Show the IFS
48   hide_ifs=False,
49
50   # Hide / Show the Annotation
51   show_ann=True,
52
53   # Hide / Show the 'points'
54   showverts=True,
55
56   # Hide / Show edges if 'points' are ordered
```

```
  showedges=False,

  # Hide / Show labels
  showlabels=False)

  # CLASS METHODS

  def init_default(self, ax):
  def set_animated(self, value):
  def create_annotations(self, ax):
  def set_visible_annotations(self, value):
  def set_animated_annotations(self, value):
  def set_data_annotations(self, positions):
  def set_data_annotations_single(self, idx, pos):
  def set_fontsize_annotations(self, fontsize):
  def set_zorder_annotations(self, zorder):
  def draw_annotations(self, ax):

# CLASS
class PropertiesIFS(PropertiesAnnotations):

  def save_to_json(self, json_path):
  def save_to_json_default(self, event):

  @classmethod
  def from_json(
  cls,
  json_path,
  ax,
  bins,
  rotation,
  color=None,
  marker=None,
  alpha=None,
  markersize=None):

  # CLASS METHODS
  def get_data(self):
  def set_data(self, mus, nus):
  def get_data_pair(self):
  def get_color(self):
  def set_color(self):
  def get_markersize(self):
  def set_markersize(self,size):
  def draw_holder_annotations(self, ax):
```

```
####  @@@@@@@@@@  @@@@@@@@@@@@@@@@@@@@@@  @@@@@@@@@@@@@@@
####   @@@@@@@@@@  widgets_operator.py   @@@@@@@@@@@@@@@
####  @@@@@@@@@@  @@@@@@@@@@@@@@@@@@@@@@  @@@@@@@@@@@@@@@
```

```
@Define the widgets for the interactive plot
widgets_operator.py

  #Gives the structure of the Widgets

  class WidgetsSimple(
  canvas=None,       # Canvas
  active_prop=None) # Active IFS property

  # Gives functionalities of the operators defined by the User

  class WidgetsSimpleOperator(WidgetsSimple) (
  canvas=None,       # Canvas
  active_prop=None) # Active IFS property

# Widgets with choice of active ifs

class WidgetsBasic(WidgetsSimple)
```

4.5 Modal operators tool

In a similar way as in the previous section, we present here the four main modal operators tool. Let us, for example fix,

$$\alpha = 0.25, \beta = 0.3$$

Next visualizations, as shown on Fig. 26 and Fig. 27, can be reproduced by the command: **python3 ifs_modal_modeler.py**

- original IFS: dark blue color
- $\mathcal{G}_{\alpha,\beta}$: orange color
- $\mathcal{F}_{\alpha,\beta}$: yellow color
- $\mathcal{H}_{\alpha,\beta}$: cyan color
- $\mathcal{J}_{\alpha,\beta}$: green color

The user can interactively drag and drop the dark blue points in the triangle corresponding to the original IFSs. By pressing the button 'Save IFS' all the results are are saved in .json formats. That is, in the folder **.\working_dir** we can see the following files:

- **ifs_0000_original_.json**
- **ifs_0000_G_.json**
- **ifs_0000_F_.json**
- **ifs_0000_H_.json**
- **ifs_0000_J_.json**

Then in our example the files corresponding to the first two files will look like,

ifs_0000_original_.json


```
{
  "data": {
    "0": [
    0.5853959296275679,
    0.0611768892043798
    ],
    "1": [
    0.00011437481734488664,
    0.6976674273681602
    ],
    "2": [
    0.0923385947687978,
    0.853244109182887
    ],
    "3": [
    0.1862602113776709,
    0.6544392729569523
    ],
    "4": [
    0.3546905113375116,
    0.4637343027513148
    ]
  },
  "contrast": 1,
  "labels_size": 12,
  "color": "blue",
  "marker_size": 0.01,
  "label": "ifs_0000_original"
}
```


ifs_0000_G_.json


```
{
  "data": {
    "0": [
    0.43904694722067594,
    0.04282382244306586
    ],
    "1": [
    8.578111300866498e-05,
    0.48836719915771215
    ],
    "2": [
    0.06925394607659835,
    0.5972708764280208
    ],
    "3": [
    0.13969515853325318,
```

```
    0.45810749106986653
    ],
    "4": [
    0.2660178835031337,
    0.3246140119259203
    ]
  },
  "contrast": 0.8,
  "labels_size": 12,
  "color": "orange",
  "marker_size": 0.01,
  "label": "ifs_0000_G"
}
```

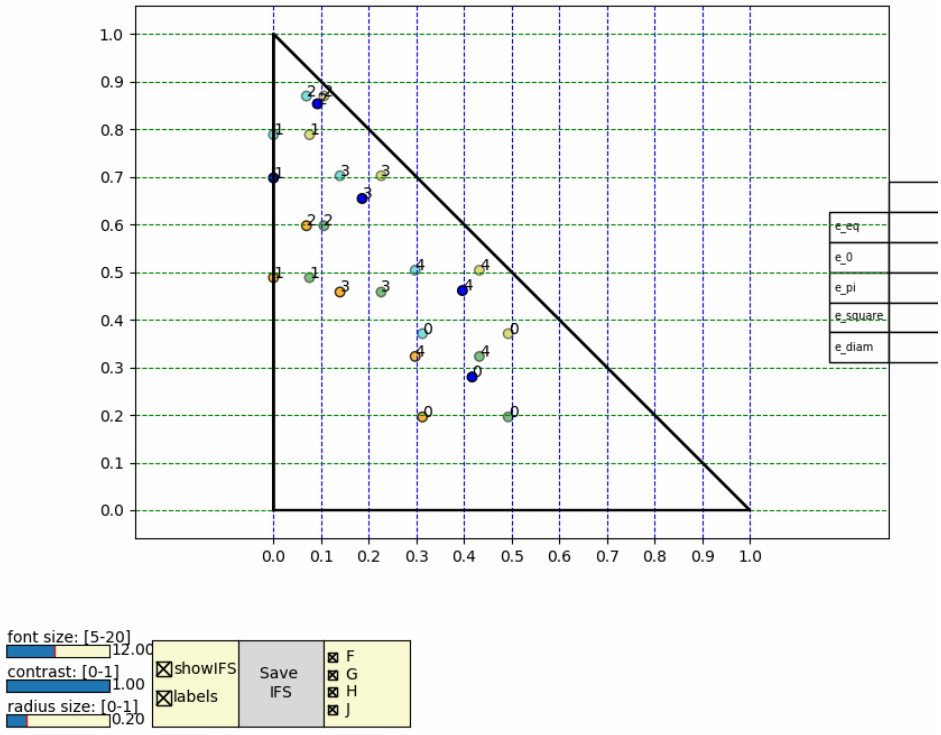
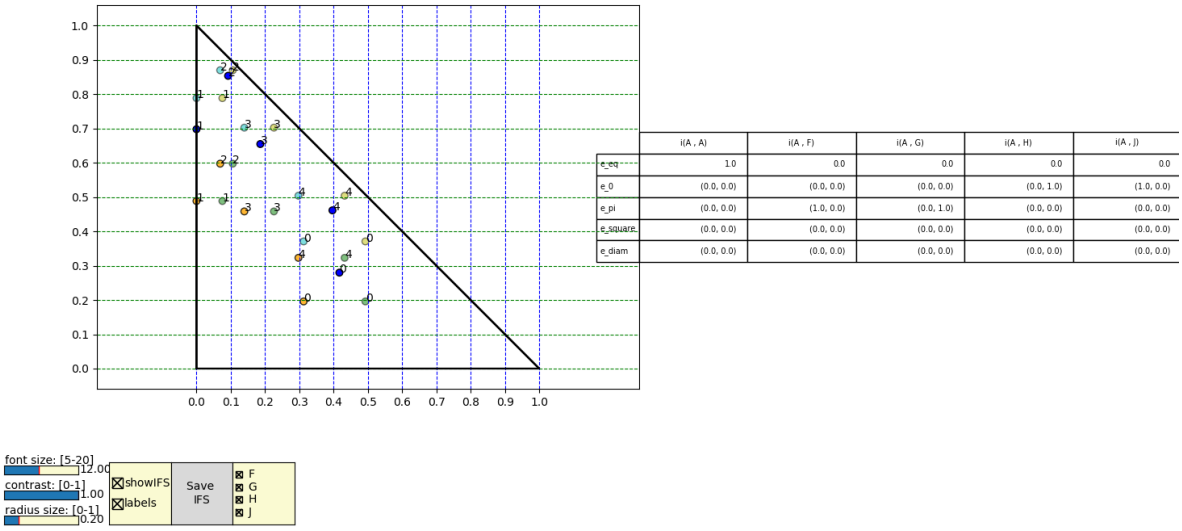



Figure 26. An interactive modeler of the four main modal operators for IFSs.



	I(A , A)	I(A , F)	I(A , G)	I(A , H)	I(A , J)
e_eq	1.0	0.0	0.0	0.0	0.0
e_0	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.0, 1.0)	(1.0, 0.0)
e_pi	(0.0, 0.0)	(1.0, 0.0)	(0.0, 1.0)	(0.0, 0.0)	(0.0, 0.0)
e_square	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)
e_diam	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)	(0.0, 0.0)

Figure 27. An interactive modeler of the four main modal operators for IFSs and the measures of their inclusion indicator compared to the original IFS.

5 Conclusion

The for computation and visualization of IFSs and their operators, proposed in this paper implementation, can be used for the modeling of real world problems within the framework of IFS. These operators can also be applied to the results from intercriteria analysis approach in order to obtain better visual feedback regarding their inherent topological aspects. Furthermore, in the future, we plan to discontinue matplotlib as the main visualization library by adopting more sophisticated and flexible tools to allow better interactivity between the user and the software.

Acknowledgements

This research has been supported by the Bulgarian National Science Fund under Grant Ref. No. KP-06-N22-1/2018 “Theoretical research and applications of InterCriteria Analysis”.

References

[1] Angelova, N. (2021). IFSTOOL – Software for Intuitionistic Fuzzy Sets Necessity, Possibility and Circle Operators. *In: Atanassov, K. et al. (eds) Uncertainty and Imprecision in Decision Making and Decision Support: New Challenges, Solutions and Perspectives. IWIFSGN 2018. Advances in Intelligent Systems and Computing*, 1081, 76–81.

[2] Atanassov, K. (1983). Intuitionistic fuzzy sets. *VII ITKR's Session, Sofia, June 1983 (Deposed in Central Sci.-Techn. Library of Bulg. Acad. of Sci., 1697/84)* (in Bulgarian). Reprinted: *International Journal Bioautomation*, 2016, 20(S1), S1–S6.

[3] Atanassov, K. (1989). Geometrical Interpretation of the Elements of the Intuitionistic Fuzzy Objects. *Mathematical Foundations of Artificial Intelligence Seminar*, Sofia, 1989, Preprint IM-MFAIS-1-89. Reprinted: *International Journal Bioautomation*, 2016, 20(S1), S27–S42.

[4] Atanassov, K. (1999). *Intuitionistic Fuzzy Sets: Theory and Applications*. Springer-Verlag, Heidelberg.

[5] Atanassov, K. (2004). On the modal operators defined over intuitionistic fuzzy sets. *Notes on Intuitionistic Fuzzy Sets*, 10(1), 7–12.

[6] Atanassov, K. (2005). On one type of intuitionistic fuzzy modal operators. *Notes on Intuitionistic Fuzzy Sets*, 11(5), 24–28.

[7] Atanassov, K. (2008). The most general form of one type of intuitionistic fuzzy modal operators. Part 2. *Notes on Intuitionistic Fuzzy Sets*, 14(1), 27–32.

[8] Atanassov, K. (2010). On two topological operators over intuitionistic fuzzy sets. *Issues in Intuitionistic Fuzzy Sets and Generalized Nets*, 8, 1–7.

[9] Atanassov K. (2012). *On Intuitionistic Fuzzy Sets Theory*. Springer-Verlag, Berlin.

[10] Atanassov, K., & Ban, A. (2000). On an operator over intuitionistic fuzzy sets. *Comptes Rendus de l'Academie bulgare des Sciences*, 53(5), 39–42.

[11] Atanassova, V., (2015). Interpretation in the Intuitionistic Fuzzy Triangle of the Results, Obtained by the InterCriteria Analysis. *Proc. of 16th World Congress of the International Fuzzy Systems Association (IFSA), 9th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT)*, 30 June – 3 July 2015, Gijon, Spain, 1369–1374.

[12] Atanassova, V., Vardeva, I., Sotirova, E., & Doukovska, L. (2016). Traversing and ranking of elements of an intuitionistic fuzzy set in the intuitionistic fuzzy interpretation triange. In: *Novel Developments in Uncertainty Representation and Processing, Vol. 401*, Advances in Intelligent Systems and Computing, Springer, 161–174.

[13] Birkhoff, G. (1967). *Lattice Theory*. American Mathematical Society, Providence, Rhode Island.

[14] Goguen, J. A. (1967). L-fuzzy sets. *Journal of Mathematical Analysis and Applications*, 18, 145–174.

[15] Marinov, E. (2014). π-ordering and index of indeterminacy for intuitionistic fuzzy sets. *Proc. of 12th Int. Workshop on IFS and GN, IWIFSGN'13, Warsaw, Oct. 2013. "Modern Approaches in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics. Volume I: Foundations"*, IBS PAN-SRI PAS, Warsaw, 129–138.

[16] Marinov, E., & Atanassov, K. (2020). Partially Continuous Pretopological and Topological Operators for Intuitionistic Fuzzy Sets. *Iranian Journal of Fuzzy Systems*, 17(2), 1–15.

[17] Marinov, E., Tsvetkov, R., & Vassilev, P. (2016). Intuitionistic Fuzzy Inclusion Indicator of Intuitionistic Fuzzy Sets. *In: Angelov P., Sotirov S. (eds) Imprecision and Uncertainty in Information Representation and Processing. Studies in Fuzziness and Soft Computing*, 332, 41–53.

[18] Mavrov, D., Radeva, I., Atanassov, K., Doukovska, L., & Kalaykov, I. (2015). InterCriteria Software Design: Graphic Interpretation within the Intuitionistic Fuzzy Triangle. *Proceedings of the 5th International Symposium on Business Modeling and Software Design (BMSD)*, 6–8 July 2015, 279–283.

[19] Vassilev, P. (2010). A Note on the Extended Modal Operator $\mathcal{G}_{\alpha,\beta}$. *Notes on Intuitionistic Fuzzy Sets*, 16(2), 12–15.

[20] Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8, 338–353.