Selection of neutral networks using q-rung neutrosophic vague soft set TOPSIS aggregating operator setting multi-criteria group decision making

A. Priya[1], P. Maragatha Meenakshi[2], Aiyared Iampan[3,*], N. Rajesh[4], Suganthi Mariyappan[5]
[1]Department of Mathematics, Government Arts College (affiliated to Bharathidasan University), Thanthonimalai, Karur 639005, Tamilnadu, India
[2]Department of Mathematics, Thanthai Periyar Government Arts and Science College (affiliated to Bharathidasan University), Tiruchirappalli 624024, Tamilnadu, India
[3]Fuzzy Algebras and Decision-Making Problems Research Unit, School of Science, University of Phayao, 19 Moo 2, Tambon Mae Ka, Amphur Mueang, Phayao 56000, Thailand
[4,5]Department of Mathematics, Rajah Serfoji Government College, Thanjavur 613005, Tamilnadu, India
Emails:a.priya@gackarur.ac.in[1]; maragathameenakship@gmail.com[2]; aiyared.ia@up.ac.th[3]; nrajesh_topology@yahoo.co.in[4]; sherin.sugan@gmail.com[5]

Abstract

The q-rung neutrosophic vague soft set (q-rung NVSS) is a generalization of the neutrosophic vague soft set (NVSS) and the vague soft set (VSS). The TOPSIS aggregated operation (AO) was used to discuss the q-rung NVSS. As an extension of VSS, the TOPSIS method effectively makes multi-criteria group decision making (MCGDM). With a score function, the goal is to find a positive and negative ideal solution based on q-rung NVSS. Closeness values are determined by presenting optimal alternatives. We provide practical examples to support our conclusions. This results in the outcome of the models for which q is provided. Considering the validity and usefulness of the models under consideration can be achieved by comparing them with those that have been proposed. Recent discoveries have generated quite a bit of interest and fascination.

Keywords: q-rung NVSS; MCGDM; TOPSIS; aggregation operator.

1 Introduction

As real-world systems become increasingly complex, decision-makers have difficulty identifying the optimal solution. Although deciding between alternatives is difficult, finding the best option is possible. A number of firms find it challenging to create opportunities, objectives, and viewpoint constraints. In order to achieve multiple objectives simultaneously, individuals or groups should consider multiple priorities when decision making (DM). Every day, we deal with a wide range of MADM-related issues. Thus, it is necessary to improve DM abilities. Various methods have been used by researchers to study this area of study. Multicriteria decision-making (MCDM) involves aggregation operators and sine operational laws. Laws of sine function symmetry and periodicity can be exploited in MCDM scenarios. Logarithmic operational laws can be incorporated into the aggregation process to realize several benefits. Firstly, sine functions are periodic, so they can be used to evaluate cyclical or recurring criteria. Decision contexts that involve cycles or seasons or decision criteria that depend on time are especially adept at utilizing this utility. In addition, symmetric properties of sine functions can be utilized in scenarios in which both negative and positive influences are equally weighted. Using symmetric aggregation operators, favorable and unfavorable outcomes can be assessed appropriately.

https://doi.org/10.54216/IJNS.220406
Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

MCDM allows effective aggregation of multiple criteria using sine operational laws-derived aggregation operators. By capturing the inherent characteristics and relationships between the criteria, we will be able to generate DM with greater accuracy and insight.

Zadeh[28] has proposed the fuzzy set (FS) as a way of dealing with uncertainties, Atanassov[2] has proposed the intuitionistic fuzzy set (IFS), Yager[27] has proposed Pythagorean fuzzy set (PFS), and Smarandache[25] proposed the neutrosophic set (NSS). There is an FS in which each element in the set has a membership value (MV) corresponding to its level of belongingness, with grades corresponding to these levels. Later, Atanassov[2] discussed the IFS, in which the sum of MV and non-membership value (NMV) does not exceed one. In the case of MV and NMV sums that exceed one, we might have difficulty DM. For generalizing IFS, Yager[27] has proposed PFS, which requires the square sum of its MV and NMV not to exceed one. Peng et al.[24] discussed the interval-valued PFS (IVPFS) based on AO. The concept of a picture fuzzy set (PicFS) was developed by Cuong et al.[9] Therefore, it has been noted that PicFS is an expanded version of IFS that can accommodate additional ambiguities. In PicFS, it was observed that MV κ, neutral ρ and NMV σ with $0 \leq \kappa+\rho+\sigma \leq 1$; for $\kappa, \rho, \sigma \in [0, 1]$. Using the PicFS definition such as "yes", "abstain", "no", and "refusal" while also avoiding missing evaluation details and encouraging the consistency of the acquired information between the actual decision environment and the evaluation data. There are many applications and studies of PicFS, but its concept has not been widely studied. Peng et al.[24] interacted with an interval-valued PFS with the AOs. The square root FS (SRFS) and its weighted AOs were discussed by Shami et al.[1] The q-rung orthopair FS (q-rung OFS) was developed by Yager[26] through an expansion of PFSs. The result of the q^{th} power of MV and q^{th} power of NMV which lies between $[0, 1]$. Since $q = 1$, q-rung OFS convert into IFSs and $q = 2$, q-rung OFS convert into PFSs, q-rung OFSs are extensions of IFSs and PFSs. A q-rung IVFSs have been discussed by Bhagawati et al.[12] Yager[26] introduced the notion of generalized OFSs.

Smarandache developed the concept of neutrosophic set (NSS).[25] This neutrality is referred to as neurosophy, and it is this neutrality that distinguishes FS from IFS. According to the truth value (TV), the indeterminacy value (IV) and the false value (FV). NSS has levels of TV, IV and FV for every component of the universe that are between $[0, 1]$. Historically, a classical set, a functional set, an integral set, etc., are generalized by an NSS. Pythagorean NSIV set (PyNSIVS) was introduced by Smarandache et al.[11] Ejegwa[10] discussed distance measures for IFSs using hamming distances (HDs), euclidean distances (EDs) and normalized Euclidean distances (NEDs). In several studies,[16–23] Palanikumar et al. studied many algebraic structures and their applications. Gau et al.[3] introduced the concept of vague sets (VSs). VS represents the two functions, such as truth membership value (TMV) T_v and false membership value (FMV) F_v. Suppose that $T_v(x)$ is the total likelihood estimate of x, derived from the evidence for x and $F_v(x)$ is the total likelihood estimate for x derived from the evidence against x. It can be noted that these functions fall into the interval $[0, 1]$, where their sum does not exceed one. Broumi et al. have explored the idea of an interval-valued Fermatean neutrosophic sets and applied it to graphs, as seen in.[4–8]

Molodtsov[15] developed the concept of soft set (SS). An SS represents real-world DM more accurately than other uncertain theories in terms of objectivity and complexity. Typically, these concepts are identified as fuzzy soft sets (FSS)[13] and intuitionistic fuzzy soft sets (IFSS).[14] There are a number of DM problems that can be addressed with these two theories. Zulqarnain et al. TOPSIS can be extended to interval-valued IFSS (IVIFSS). Using TOPSIS, distances to positive ideal solutions (PIS) and negative ideal solutions (NIS) are calculated, and a preference order is found based on the relative closeness of the two distance measures.

Consequently, this work makes the following contributions:

1. TOPSIS q-rung NVSS introduced a new ED measure.
2. The proposed definition of NVSS with a q-rung is applied.
3. As a result of the q-rung NVSS, PISs and NISs are determined.
4. The result is determined based on q.

The contribution of this research extends the concept of TOPSIS using q-rung NVSS. The paper is divided into six sections. The introduction is found in section 1. A q-rung VS and q-rung IVFS are provided in Section 2. Section 3 discusses about MCGDM using q-rung NVSS algorithm. Section 4 presents a real-life example using q-rung NVSS-TOPSIS aggregating operator. Section 5 discusses the comparison for the q-rung NVSS-TOPSIS and existing approach. Section 6 provides a conclusion.

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

2 Preliminaries

This part aims to review some ideas related to NSS and q-rung FS literary concepts.

Definition 2.1. A NSS ζ in the universe $\mathbb{U}$ is $\zeta = \{\tau, \Xi_\zeta^T(\tau), \Xi_\zeta^I(\tau), \Xi_\zeta^F(\tau) | \tau \in \mathbb{U}\}$, where $\Xi_\zeta^T(\tau)$, $\Xi_\zeta^I(\tau)$ $\Xi_\zeta^F(\tau)$ represents the TV, IV, and FV of ζ, respectively. Then $\Xi_\zeta^T, \Xi_\zeta^I, \Xi_\zeta^F : \mathbb{U} \to [0,1]$ and $0 \leq \sup \Xi_\zeta^T(\tau) + \sup \Xi_\zeta^I(\tau) + \sup \Xi_\zeta^F(\tau) \leq 3$.

Definition 2.2. [26] The q-rung FS ζ in $\mathbb{U}$ is $\zeta = \left\{\tau, \left\langle \Xi_\zeta^T(\tau), \Xi_\zeta^F(\tau) \right\rangle \middle| \tau \in \mathbb{U}\right\}$, $\Xi_\zeta^T : \mathbb{U} \to [0,1]$ and $\Xi_\zeta^F : \mathbb{U} \to [0,1]$ denotes the MV and NMV of $\tau \in \mathbb{U}$ to ζ, respectively and $0 \leq (\Xi_\zeta^T(\tau))^q + (\Xi_\zeta^F(\tau))^q \leq 1$, where $q \succeq 1$. The degree of indeterminacy is $\pi(\tau) = \left((\Xi_\zeta^T(\tau))^q + (\Xi_\zeta^F(\tau))^q - (\Xi_\zeta^T(\tau))^q (\Xi_\zeta^F(\tau))^q\right)^{1/q}$. For $\zeta = \left\langle \Xi_\zeta^T, \Xi_\zeta^F \right\rangle$ is called a q-rung FN.

Definition 2.3. [12] The q-rung IVFS ζ in $\mathbb{U}$ is $\zeta = \left\{\tau, \left\langle \widehat{\Xi_\zeta^T}(\tau), \widehat{\Xi_\zeta^F}(\tau) \right\rangle \middle| \tau \in \mathbb{U}\right\}$, where $\widehat{\Xi_\zeta^T} : \mathbb{U} \to Int([0,1])$ and $\widehat{\Xi_\zeta^F} : \mathbb{U} \to Int([0,1])$ denote the MV and NMV of $\tau \in \mathbb{U}$ to ζ, respectively, and $0 \leq (\Xi_\zeta^{T+}(\tau))^q + (\Xi_\zeta^{F+}(\tau))^q \leq 1$. For convenience, $\zeta = \left\langle \left[\Xi_\zeta^{T-}, \Xi_\zeta^{T+}\right], \left[\Xi_\zeta^{F-}, \Xi_\zeta^{F+}\right] \right\rangle$ is called a q-rung IVFN.

Definition 2.4. [11] A PyNSS ζ in $\mathbb{U}$ is $\zeta = \{\tau, \Xi_\zeta^T(\tau), \Xi_\zeta^I(\tau), \Xi_\zeta^F(\tau) | \tau \in \mathbb{U}\}$, where $\Xi_\zeta^T(\tau)$, $\Xi_\zeta^I(\tau)$ $\Xi_\zeta^F(\tau)$ represents the TV, IV and FV of ζ, respectively. The mapping $\Xi_\zeta^T, \Xi_\zeta^I, \Xi_\zeta^F : \mathbb{U} \to [0,1]$ and $0 \leq (\Xi_\zeta^T(\tau))^2 + (\Xi_\zeta^I(\tau))^2 + (\Xi_\zeta^F(\tau))^2 \leq 2$. Since $\{\zeta = \langle \Xi_\zeta^T, \Xi_\zeta^I, \Xi_\zeta^F \rangle$ is called a Pythagorean neutrosophic number (PyNSN).

Definition 2.5. The PyIVFS $\zeta = \left\{\tau, \left\langle \widehat{\Xi_\zeta^T}(\tau), \widehat{\Xi_\zeta^F}(\tau) \right\rangle \middle| \tau \in \mathbb{U}\right\}$, where $\widehat{\Xi_\zeta^T}(\tau) = \left[\Xi_\zeta^{Tl}(\tau), \Xi_\zeta^{Tu}(\tau)\right]$ and $\widehat{\Xi_\zeta^F}(\tau) = \left[\Xi_\zeta^{Fl}(\tau), \Xi_\zeta^{Fu}(\tau)\right]$ denotes the MV and NMV of ζ, respectively. Here $\widehat{\Xi_\zeta^T}$ and $\widehat{\Xi_\zeta^F}$ are function from $\mathbb{U}$ into $\mathcal{D}[0,1]$ and $0 \leq (\widehat{\Xi_\zeta^T}(\tau))^2 + (\widehat{\Xi_\zeta^F}(\tau))^2 \leq 1$ it is observed that $0 \leq (\Xi_\zeta^{Tu}(\tau))^2 + (\Xi_\zeta^{Fu}(\tau))^2 \leq 1$.

Definition 2.6. The IVNSS $\zeta = \left\{\tau, \left(\widehat{\Xi_\zeta^T}(\tau), \widehat{\Xi_\zeta^I}(\tau), \widehat{\Xi_\zeta^F}(\tau)\right) | \tau \in \mathbb{U}\right\}$, where $\widehat{\Xi_\zeta^T}(\tau) = \left[\Xi_\zeta^{Tl}(\tau), \Xi_\zeta^{Tu}(\tau)\right]$, $\widehat{\Xi_\zeta^I}(\tau) = \left[\Xi_\zeta^{Il}(\tau), \Xi_\zeta^{Iu}(\tau)\right]$ and $\widehat{\Xi_\zeta^F}(\tau) = \left[\Xi_\zeta^{Fl}(\tau), \Xi_\zeta^{Fu}(\tau)\right]$ represents the TV, IV, and FV of ζ, respectively. Then $\widehat{\Xi_\zeta^T} : \mathbb{U} \to D[0,1]$, $\widehat{\Xi_\zeta^I} : \mathbb{U} \to D[0,1]$, $\widehat{\Xi_\zeta^F} : \mathbb{U} \to D[0,1]$ and $0 \leq (\widehat{\Xi_\zeta^T}(\tau))^2 + (\widehat{\Xi_\zeta^I}(\tau))^2 + (\widehat{\Xi_\zeta^F}(\tau))^2 \leq 2$ mean $0 \leq (\Xi_\zeta^{Tu}(\tau))^2 + (\Xi_\zeta^{Iu}(\tau))^2 + (\Xi_\zeta^{Fu}(\tau))^2 \leq 2$.
Here $\widehat{\zeta} = \left(\left[\Xi_\zeta^{Tl}, \Xi_\zeta^{Tu}\right], \left[\Xi_\zeta^{Il}, \Xi_\zeta^{Iu}\right], \left[\Xi_\zeta^{Fl}, \Xi_\zeta^{Fu}\right]\right)$ is called a neutrosophic interval-valued number (NSIVN).

Definition 2.7. Let E be the set of parameters. The $\widehat{(\Gamma, \zeta)}$ or $\widehat{\Gamma_\zeta}$ is called a NSIVS on $\mathbb{U}$ if $\zeta \sqsubseteq E$ and $\Gamma : \zeta \to NSIV^{\mathbb{U}}$, where $NSIV^{\mathbb{U}}$ is denote the set of all neutrosophic interval-valued subsets of $\mathbb{U}$. That is,
$$\widehat{\Gamma_\zeta} = \left\{ \left(e, \left\{ \frac{\tau}{\left(\left[\Xi_{\Gamma_\zeta}^{Tl}(\tau), \Xi_{\Gamma_\zeta}^{Tu}(\tau)\right], \left[\Xi_{\Gamma_\zeta}^{Il}(\tau), \Xi_{\Gamma_\zeta}^{Iu}(\tau)\right], \left[\Xi_{\Gamma_\zeta}^{Fl}(\tau), \Xi_{\Gamma_\zeta}^{Fu}(\tau)\right]\right)} \right\} \right) \middle| e \in \zeta, \tau \in \mathbb{U} \right\}.$$

Remark 2.8. If we write $\widehat{\kappa_{ij}} = \widehat{\Xi_{\Gamma_\zeta}^T}(e_j)(\tau_i)$, $\widehat{\rho_{ij}} = \widehat{\Xi_{\Gamma_\zeta}^I}(e_j)(\tau_i)$ and $\widehat{\sigma_{ij}} = \widehat{\Xi_{\Gamma_\zeta}^F}(e_j)(\tau_i)$, where $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$ then the NSIV set $\widehat{\Gamma_\zeta}$ defined in matrix form:

$$\widehat{\Gamma_\zeta} = [(\widehat{\kappa_{ij}}, \widehat{\rho_{ij}}, \widehat{\sigma_{ij}})]_{m \times n} = \begin{bmatrix} (\widehat{\kappa_{11}}, \widehat{\rho_{11}}, \widehat{\sigma_{11}}) & (\widehat{\kappa_{12}}, \widehat{\rho_{12}}, \widehat{\sigma_{12}}) & \dots & (\widehat{\kappa_{1n}}, \widehat{\rho_{1n}}, \widehat{\sigma_{1n}}) \\ (\widehat{\kappa_{21}}, \widehat{\rho_{21}}, \widehat{\sigma_{21}}) & (\widehat{\kappa_{22}}, \widehat{\rho_{22}}, \widehat{\sigma_{22}}) & \dots & (\widehat{\kappa_{2n}}, \widehat{\rho_{2n}}, \widehat{\sigma_{2n}}) \\ \vdots & \vdots & \ddots & \vdots \\ (\widehat{\kappa_{m1}}, \widehat{\rho_{m1}}, \widehat{\sigma_{m1}}) & (\widehat{\kappa_{m2}}, \widehat{\rho_{m2}}, \widehat{\sigma_{m2}}) & \dots & (\widehat{\kappa_{mn}}, \widehat{\rho_{mn}}, \widehat{\sigma_{mn}}) \end{bmatrix}$$

This matrix is called a neutrosophic interval-valued soft matrix (NSIVSM).

Definition 2.9. [27] Let $\mathbb{X}$ be the universe, PFS ζ in $\mathbb{X}$ is $\zeta = \left\{\epsilon, \left\langle \Xi_\zeta^T(\epsilon), \Xi_\zeta^F(\epsilon) \right\rangle \middle| \epsilon \in \mathbb{X}\right\}$, $\Xi_\zeta^T : \mathbb{X} \to [0,1]$ and $\Xi_\zeta^F : \mathbb{X} \to [0,1]$ are represents the MD and NMD of $\epsilon \in \mathbb{X}$ to ζ, respectively and $0 \leq (\Xi_\zeta^T(\epsilon))^2 + (\Xi_\zeta^F(\epsilon))^2 \leq 1$. For, $\zeta = \left\langle \Xi_\zeta^T, \Xi_\zeta^F \right\rangle$ is called the Pythagorean fuzzy number (PyFN).

Definition 2.10. [1] The SRFS ζ in $\mathbb{X}$ is $\zeta = \left\{\epsilon, \left\langle \Xi_\zeta^T(\epsilon), \Xi_\zeta^F(\epsilon) \right\rangle \middle| \epsilon \in \mathbb{X}\right\}$, $\Xi_\zeta^T : \mathbb{X} \to [0,1]$ and $\Xi_\zeta^F : \mathbb{X} \to [0,1]$ are represents the MD and NMD of $\epsilon \in \mathbb{X}$ to ζ, respectively and $0 \leq (\Xi_\zeta^T(\epsilon))^2 + \sqrt{\Xi_\zeta^F(\epsilon)} \leq 1$. For, $\zeta = \left\langle \Xi_\zeta^T, \Xi_\zeta^F \right\rangle$ is called the square root fuzzy number (SRFN).

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

Definition 2.11. [24] The PyIVFS ζ in $\mathbb{X}$ is $\widehat{\zeta} = \left\{\epsilon, \left\langle \widehat{\Xi^T_\zeta}(\epsilon), \widehat{\Xi^F_\zeta}(\epsilon)\right\rangle \middle| \epsilon \in \mathbb{X}\right\}$, where $\widehat{\Xi^T_\zeta} : \mathbb{X} \to [0,1]$ and $\widehat{\Xi^F_\zeta} : \mathbb{X} \to Int([0,1])$ are represents the MD and NMD of $\epsilon \in \mathbb{X}$ to ζ, respectively, and $0 \leq (\Xi^{Tu}_\zeta(\epsilon))^2 + (\Xi^{Fu}_\zeta(\epsilon))^2 \leq 1$. For, $\widehat{\zeta} = \left\langle \left[\Xi^{Tl}_\zeta, \Xi^{Tu}_\zeta\right], \left[\Xi^{Fl}_\zeta, \Xi^{Fu}_\zeta\right]\right\rangle$ is called a Pythagorean interval-valued fuzzy number (PyIVFN).

Definition 2.12. [3] (i) A VS ζ_1 in $\mathbb{X}$ is a pair $(T_{\zeta_1}, F_{\zeta_1})$, $T_{\zeta_1} : \mathbb{X} \to [0,1]$, $F_{\zeta_1} : \mathbb{X} \to [0,1]$ and $T_{\zeta_1}(\epsilon) + F_{\zeta_1}(\epsilon) \leq 1$, for all $\epsilon \in \mathbb{X}$, T_{ζ_1} and F_{ζ_1} are called true membership and false membership function, respectively.
(ii) $\zeta_1(\epsilon) = [T_{\zeta_1}(\epsilon), 1 - F_{\zeta_1}(\epsilon)]$ is represent the vague value of ϵ in ζ_1.

Definition 2.13. [3] (i) A VS ζ_1 is contained in the other VS ζ_2, $\zeta_1 \subseteq \zeta_2$ if and only if $\zeta_1(\epsilon) \leq \zeta_{1\zeta_2}(\epsilon)$. That is, $T_{\zeta_1}(\epsilon) \leq T_{\zeta_2}(\epsilon)$ and $1 - F_{\zeta_1}(\epsilon) \leq 1 - F_{\zeta_2}(\epsilon)$, for all $\epsilon \in \mathbb{X}$.
(ii) The union of two VSs ζ_1 and ζ_2, as $X = \zeta_1 \cup \zeta_2$, $T_X = \max\{T_{\zeta_1}, T_{\zeta_2}\}$ and $1 - F_X = \max\{1 - F_{\zeta_1}, 1 - F_{\zeta_2}\} = 1 - \min\{F_{\zeta_1}, F_{\zeta_2}\}$.
(iii) The intersection of two VSs ζ_1 and ζ_2 as $X = \zeta_1 \cap \zeta_2$, $T_X = \min\{T_{\zeta_1}, T_{\zeta_2}\}$ and $1 - F_X = \min\{1 - F_{\zeta_1}, 1 - F_{\zeta_2}\} = 1 - \max\{F_{\zeta_1}, F_{\zeta_2}\}$.

Definition 2.14. [3] A VS ζ_1 of a set $\mathbb{X}$, for all $\epsilon \in \mathbb{X}$. Then
(i) $T_{\zeta_1}(\epsilon) = 0$ and $F_{\zeta_1}(\epsilon) = 1$ is called zero VS of $\mathbb{X}$.
(ii) $T_{\zeta_1}(\epsilon) = 1$ and $F_{\zeta_1}(\epsilon) = 0$ is called a unit VS of $\mathbb{X}$.

3 MCGDM based on q-rung NVSS-TOPSIS aggregating operator

Definition 3.1. The cardinal set of the q-rung NVSS $\widehat{\Gamma_X}$ is denoted by $\widehat{c\Gamma_X}$ and is defined as

$$\widehat{c\Gamma_X} = \left\{ \frac{e}{\left(\left[\Xi^{Tl}_{c\alpha_X}(e), \Xi^{Tu}_{c\alpha_X}(e)\right], \left[\Xi^{Il}_{c\beta_X}(e), \Xi^{Iu}_{c\beta_X}(e)\right], \left[\Xi^{1-Tu}_{c\gamma_X}(e), \Xi^{1-Tl}_{c\gamma_X}(e)\right]\right)} \middle| e \in E \right\}$$

$$= \left\{ \frac{e}{\left(\widehat{\Xi^T}_{c\alpha_X}(e), \widehat{\Xi^I}_{c\beta_X}(e), \widehat{\Xi^{1-T}}_{c\gamma_X}(e)\right)} \middle| e \in E \right\}, \text{ where } \widehat{\Xi^T}_{c\alpha_X}, \widehat{\Xi^I}_{c\beta_X} \text{ and } \widehat{\Xi^{1-T}}_{c\gamma_X} : E \to D[0,1] \text{ are mapping, respectively, where } \widehat{\Xi^T}_{c\alpha_X}(e) = \frac{|\widehat{\alpha}_X(e)|}{|\mathbb{U}|}, \widehat{\Xi^I}_{c\beta_X}(e) = \frac{|\widehat{\beta}_X(e)|}{|\mathbb{U}|} \text{ and } \widehat{\Xi^F}_{c\gamma_X}(e) = \frac{|\widehat{\gamma}_X(e)|}{|\mathbb{U}|}, \text{ where } |\widehat{\alpha}_X(e)|, |\widehat{\beta}_X(e)| \text{ and } |\widehat{\gamma}_X(e)| \text{ denote the scalar cardinalities of the } q\text{-rung NVSS } \widehat{\alpha}_X(e), \widehat{\beta}_X(e) \text{ and } \widehat{\gamma}_X(e) \text{ respectively, and } |\mathbb{U}| \text{ represents cardinality of the universe } \mathbb{U}. \text{ The collection of all cardinal sets of } q\text{-rung NVSS of } \mathbb{U} \text{ is represented as } c\text{q}-rung\ NV^{\mathbb{U}}. \text{ If } X \subseteq E = \{e_i | i = 1, 2, ..., n\}, \text{ then } \widehat{c\Gamma_X} \in c\text{q}-rung\ NVS^{\mathbb{U}} \text{ may be represented in matrix form as}$$

$$\left[\left([\kappa^l_{1j}, \kappa^u_{1j}], [\rho^l_{1j}, \rho^u_{1j}], [\sigma^l_{1j}, \sigma^u_{1j}]\right) \right]_{1\times n} = \left[\left([\kappa^l_{11}, \kappa^u_{11}], [\rho^l_{11}, \rho^u_{11}], [\sigma^l_{11}, \sigma^u_{11}]\right), \left([\kappa^l_{12}, \kappa^u_{12}], [\rho^l_{12}, \rho^u_{12}], [\sigma^l_{12}, \sigma^u_{12}]\right) \right.$$
$$\left., ..., \left([\kappa^l_{1n}, \kappa^u_{1n}], [\rho^l_{1n}, \rho^u_{1n}], [\sigma^l_{1n}, \sigma^u_{1n}]\right) \right], \text{ where } \left([\kappa^l_{1j}, \kappa^u_{1j}], [\rho^l_{1j}, \rho^u_{1j}], [\sigma^l_{1j}, \sigma^u_{1j}]\right) = \left[\mu^l_{c\Gamma_X}(e_j), \mu^u_{c\Gamma_X}(e_j)\right],$$
for $j = 1, 2, ..., n$. For our convenience, consider matrix form as $[(\widehat{\kappa}_{1j}, \widehat{\rho}_{1j}, \widehat{\sigma}_{1j})]_{1\times n} = [(\widehat{\kappa}_{11}, \widehat{\rho}_{11}, \widehat{\sigma}_{11}), (\widehat{\kappa}_{12}, \widehat{\rho}_{12}, \widehat{\sigma}_{12}), ..., (\widehat{\kappa}_{1n}, \widehat{\rho}_{1n}, \widehat{\sigma}_{1n})]$, where $(\widehat{\kappa}_{1j}, \widehat{\rho}_{1j}, \widehat{\sigma}_{1j}) = \widehat{\mu}_{c\Gamma_X}(e_j)$, for $j = 1, 2, ..., n$. Hence this matrix is called a cardinal matrix of $\widehat{c\Gamma_X}$ of E.

Definition 3.2. Let $\widehat{\Gamma_X} \in q\text{-rung NVSS}(\mathbb{U})$ and $\widehat{c\Gamma_X} \in cq\text{-rung NVSS}(\mathbb{U})$. The q-rung NVSS AO $q - rung\ NVSS_{agg} : cq - rung\ NVSS(\mathbb{U}) \times q - rungNVSS(\mathbb{U}) \to q - rungNVSS(\mathbb{U}, E)$ is defined as

$$q - rungNVSS_{agg}(\widehat{c\Gamma_X}, \widehat{\Gamma_X}) = \left\{ \frac{\tau}{\widehat{\mu}_{\Gamma^*_X}(\tau)} \middle| \tau \in \mathbb{U} \right\} = \left\{ \frac{\tau}{\left(\widehat{\Xi^T}_{\alpha^*_X}(\tau), \widehat{\Xi^I}_{\beta^*_X}(\tau), \widehat{\Xi^{1-T}}_{\gamma^*_X}(\tau)\right)} \middle| \tau \in \mathbb{U} \right\}.$$

This collection is called $q - rungNVSSset\widehat{\Gamma_X}$.
The TD $\widehat{\Xi^T}_{\alpha^*_X}(\tau) : \mathbb{U} \to D[0,1]$ by $\widehat{\Xi^T}_{\alpha^*_X}(\tau) = \frac{1}{|E|}\sum_{e\in E}\left(\widehat{\Xi^T}_{c\alpha_X}(e), \widehat{\Xi^T}_{\alpha_X}(e)\right)(\tau)$, ID $\widehat{\Xi^I}_{\beta^*_X}(\tau) : \mathbb{U} \to D[0,1]$ by $\widehat{\Xi^I}_{\beta^*_X}(\tau) = \frac{1}{|E|}\sum_{e\in E}\left(\widehat{\Xi^I}_{c\beta_X}(e), \widehat{\Xi^I}_{\beta_X}(e)\right)(\tau)$ and FD $\widehat{\Xi^{1-T}}_{\gamma^*_X}(\tau) : \mathbb{U} \to D[0,1]$ by $\widehat{\Xi^{1-T}}_{\gamma^*_X}(\tau) = \frac{1}{|E|}\sum_{e\in E}\left(\widehat{\Xi^{1-T}}_{c\gamma_X}(e), \widehat{\Xi^{1-T}}_{\gamma_X}(e)\right)(\tau)$. The set $q - rungNVSS_{agg}(\widehat{c\Gamma_X}, \widehat{\Gamma_X})$ is expressed in matrix form as

$$\left[\left([\kappa^l_{i1}, \kappa^u_{i1}], [\rho^l_{i1}, \rho^u_{i1}], [\sigma^l_{i1}, \sigma^u_{i1}]\right) \right]_{m\times 1} = \begin{bmatrix} \left([\kappa^l_{11}, \kappa^u_{11}], [\rho^l_{11}, \rho^u_{11}], [\sigma^l_{11}, \sigma^u_{11}]\right) \\ \left([\kappa^l_{21}, \kappa^u_{21}], [\rho^l_{21}, \rho^u_{21}], [\sigma^l_{21}, \sigma^u_{21}]\right) \\ \vdots \\ \left([\kappa^l_{m1}, \kappa^u_{m1}], [\rho^l_{m1}, \rho^u_{m1}], [\sigma^l_{m1}, \sigma^u_{m1}]\right) \end{bmatrix}$$

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

where $\left[\left([\kappa_{i1}^l,\kappa_{i1}^u],[\rho_{i1}^l,\rho_{i1}^u],[\sigma_{i1}^l,\sigma_{i1}^u]\right)\right]=\left[\mu_{\Gamma_X^*}^l(\tau_i),\mu_{\Gamma_X^*}^u(\tau_i)\right]$, for $i=1,2,...,m$. This matrix is called the q-rung NVSS aggregate matrix of $q-rungNVSS_{agg}(\widehat{c\Gamma_X},\widehat{\Gamma_X})$ over $\mathbb{U}$.

Algorithm-V (q-rung NVSS-TOPSIS)

Step-1: Suppose that $\mathcal{D}=\{\mathcal{D}_i:i\in\mathbb{N}\}$ is a set of decision makers $y=\{\chi_i:i\in\mathbb{N}\}$ is a set of alternatives and $D=\{e_i:i\in\mathbb{N}\}$ is a set of parameters, where $\mathcal{D}$ is the average of $\mathcal{D}_i$.
Step-2: Determine the weighted parameter by $\mathcal{P}=\left[\varpi_{ij}^l,\varpi_{ij}^u\right]_{n\times m}$, where $\left[\varpi_{ij}^l,\varpi_{ij}^u\right]$ be the weight by the makers $\mathcal{D}_i$ to e_j.
Step-3: Determine the weighted normalized decision by $\widehat{\mathcal{N}}=\left[\widehat{n}_{ij}^l,\widehat{n}_{ij}^u\right]_{n\times m}$, where
$\left[\widehat{n}_{ij}^l,\widehat{n}_{ij}^u\right]=\left[\frac{\varpi_{ij}^l}{\sqrt[q]{\sum_{i=1}^n\varpi_{ij}^{qu}}},\frac{\varpi_{ij}^u}{\sqrt[q]{\sum_{i=1}^n\varpi_{ij}^{ql}}}\right]$ is the normalized parameter and finding the weighted vector is
$\mathcal{W}=\left([\mu_1^l,\mu_1^u],[\mu_2^l,\mu_2^2]...,[\mu_m^l,\mu_m^u]\right)$, where $[\mu_i^l,\mu_i^u]=\left[\frac{\varpi_i^l}{\sqrt[q]{\sum_{l=1}^n\varpi_{li}^u}},\frac{\varpi_i^u}{\sqrt[q]{\sum_{l=1}^n\varpi_{li}^l}}\right]$ be the weight of the j^{th} parameter and $[\varpi_j^l,\varpi_j^u]=\left[\frac{\sum_{i=1}^n\widehat{n}_{ij}^l}{n},\frac{\sum_{i=1}^n\widehat{n}_{ij}^u}{n}\right]$.
Step-4: Form the q-rung NVSS decision by $\mathcal{D}_i=\left[c_{jk}^{li},c_{jk}^{ui}\right]_{l\times m}$. Here $\left[c_{jk}^{li},c_{jk}^{ui}\right]$ is a i^{th} decision maker, i.e., $[\mathcal{D}_i^l,\mathcal{D}_i^u]$ for each i. Calculating the aggregating matrix by
$\left[\mathcal{X}^l,\mathcal{X}^u\right]=\frac{\left[\mathcal{D}_1^l,\mathcal{D}_1^u\right]+\left[\mathcal{D}_2^l,\mathcal{D}_2^u\right]+...+\left[\mathcal{D}_n^l,\mathcal{D}_n^u\right]}{n}=\left[x_{jk}^l,x_{jk}^u\right]_{l\times m}$.
Step-5: Find the weighted q-rung NVSS decision matrix by $[\mathcal{Y}^l,\mathcal{Y}^u]=\left[\chi_{jk}^l,\chi_{jk}^u\right]_{l\times m}$,
where $\left[\chi_{jk}^l,\chi_{jk}^u\right]=\left[\mu_k^l\times x_{jk}^l,\mu_k^u\times x_{jk}^u\right]$.
Step-6: Calculate the values for q-rung NVSS-PIS and q-rung NVSS-NIS. Now,
q-rung NVSS-PIS = $\left([\chi_1^{l+},\chi_1^{u+}],[\chi_2^{l+},\chi_2^{u+}],...,[\chi_l^{l+},\chi_l^{u+}]\right)$
$=\left\{\left(\vee_k[\chi_{jk}^l,\chi_{jk}^u],\wedge_k[\chi_{jk}^l,\chi_{jk}^u],\wedge_k[\chi_{jk}^l,\chi_{jk}^u]\right)|k=1,2,...,m\right\}$ and
q-rung NVSS-PIS = $\left([\chi_1^{l-},\chi_1^{u-}],[\chi_2^{l-},\chi_2^{u-}]...,[\chi_l^{l-},\chi_l^{u-}]\right)$
$=\left\{\left(\wedge_k[\chi_{jk}^l,\chi_{jk}^u],\vee_k[\chi_{jk}^l,\chi_{jk}^u],\vee_k[\chi_{jk}^l,\chi_{jk}^u]\right)|k=1,2,...,m\right\}$.
Here q-rung NVSS union $\vee$ and q-rung NVSS intersection $\wedge$.
Step-7: Obtain q-rung NVSS-Euclidean distances from q-rung NVSS-PIS and q-rung NVSS-NIS. Now
$$\left[d_j^{l+},d_j^{u+}\right]=\left[\sqrt{\sum_{k=1}^m\left\{\left(\Xi_{jk}^{Tl}-\Xi_j^{Tl+}\right)^2+\left(\Xi_{jk}^{Il}-\Xi_j^{Il+}\right)^2+\left(\Xi_{jk}^{(1-T)l}-\Xi_j^{(1-T)l+}\right)^2\right\}},\right.$$
$$\left.\sqrt{\sum_{k=1}^m\left\{\left(\Xi_{jk}^{Tu}-\Xi_j^{Tu+}\right)^2+\left(\Xi_{jk}^{Iu}-\Xi_j^{Iu+}\right)^2+\left(\Xi_{jk}^{(1-T)u}-\Xi_j^{(1-T)u+}\right)^2\right\}}\right],$$
$$\left[d_j^{l-},d_j^{u-}\right]=\left[\sqrt{\sum_{k=1}^m\left\{\left(\Xi_{jk}^{Tl}-\Xi_j^{Tl-}\right)^2+\left(\Xi_{jk}^{Il}-\Xi_j^{Il-}\right)^2+\left(\Xi_{jk}^{(1-T)l}-\Xi_j^{(1-T)l-}\right)^2\right\}},\right.$$
$$\left.\sqrt{\sum_{k=1}^m\left\{\left(\Xi_{jk}^{Tu}-\Xi_j^{Tu-}\right)^2+\left(\Xi_{jk}^{Iu}-\Xi_j^{Iu-}\right)^2+\left(\Xi_{jk}^{(1-T)u}-\Xi_j^{(1-T)u-}\right)^2\right\}}\right],$$
where $j=1,2,...,n$.
Step-8: Calculate the relative closeness for the ideal solution by
$\left[C^{l*}(\chi_j),C^{u*}(\chi_j)\right]=\left[\frac{d_j^{l-}}{d_j^{u+}+d_j^{u-}},\frac{d_j^{u-}}{d_j^{l+}+d_j^{l-}}\right]$, hence, $C^*(\chi_j)=\frac{C^{l*}(\chi_j)+C^{u*}(\chi_j)}{2}\in[0,1]$.
Step-9: Finding the rank of alternatives by using decreasing or increasing order of their relative closeness coefficients. The bigger $C^*(\chi_j)$, the more desirable alternative χ_j.

4 Selection process based on neural networks

Adapted from the structure and function of the human brain, neural networks are a type of machine learning model. It is one of the basic components of deep learning, an area of machine learning that has become

https://doi.org/10.54216/IJNS.220406
Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

increasingly popular and successful in recent years. In addition to image and speech recognition, neural networks are used for natural language processing. Biological neurons in the human brain are inspired by artificial neurons in neural networks. Synthetic neurons receive inputs, process them mathematically, and produce outputs. A neural network is composed of layers of neurons. Weights and biases are associated with each connection between neurons. A neuron's response to input is determined by these parameters, which are learned during the training process. Supervised learning is used to train neural networks. Weights and biases are adjusted during training based on the model's predictions and actual target values to minimise prediction error. Activation functions multiply input data by weights and propagate the results through layers. Computer vision, natural language processing, speech recognition, and reinforcement learning are some of the fields in which neural networks have demonstrated remarkable capabilities. Technology has advanced significantly as a result of their contributions, and they have the potential to continue to grow and innovate. As soon as the neural network has processed enough examples, it will be able to process inputs that are new and unknown, and it will be able to produce correct results. The results usually become more accurate when the program learns to identify a greater variety of inputs and instances.

1. Feedforward neural network (FNN) :(χ_1)
   Neural networks consist of three layers: an input, a hidden, and an output layer. One of the most common types of artificial neural networks is the feedforward neural network (FNN) or multilayer perceptron (MLP). There is no feedback loop or recurrent connection between the input and output layers, so it is called feedforward because information flows in only one direction. The feature or attribute of the input data is represented by each neuron in the input layer. Input data has a dimensionality determined by the number of neurons in the input layer. One or more hidden layers between the input and output layers are possible. Data input is processed by artificial neurons in these hidden layers. In other words, these layers aren't directly connected to the outside. Final results or predictions are produced by the output layer. Depending on the task, the output layer contains a certain number of neurons. Feedforward neural networks consist of neurons that perform two main functions. The weighted sum of the inputs is taken into account by neurons. Weighted inputs are added together and multiplied by a weight. As a result, a nonlinearity is introduced into the model because the weighted sum is then passed through an activation function. Each connection between neurons has a weight that represents its strength. A training process teaches these weights. The weighted sum of each neuron can be adjusted by a bias term. In the same way that weights are learned through training, biases are also acquired. The application of feedforward neural networks includes regression, binary and multiclass classification, function approximation, and pattern recognition, among others. In the field of deep learning, they are fundamental building blocks, and they can be modified and extended to create more complex neural network architectures.

2. Deconvolutional neural network (DNN) :(χ_2)
   An image segmentation, object localization, or generative image modelling task can be managed using a DNN. A DNN reverses or undoes down sampling operations performed by convolutional neural networks (CNNs), such as max-pooling and subsampling. To recover spatial information lost during downsampling, these methods perform upsampling or "deconvolution" to produce high-resolution input feature maps. The transposed convolution operation is the most common operation in a DNN. A feature map's spatial resolution is increased by this operation. The output features of transposed convolutional layers are mapped to lower-resolution input features using learnable parameters. To recover spatial information lost during max-pooling or subsampling in CNNs, DNNs often include unpooling or unsubsampling layers in addition to transposed convolution. As the downsampling process continues, unspooling layers restore information to its original locations. By incorporating skip connections, DNNs often produce higher-resolution feature maps that concatenate or add information from earlier layers to those produced by the deconvolution layers to capture local and global contexts. While keeping the context of coarser layers, these skip connections assist in refining the output's details. The output layer of a DNN may differ based on the task. There are a number of applications for DNNs in computer vision, such as semantic segmentation, object localization, super-resolution, generalized modelling, and image-to-image translation. Computer vision and image processing applications use DNNs to recover fine-grained spatial details and spatial information.

3. Recurrent neural network (RNN) :(χ_3)
   An artificial neural network (ANN), known as a recurrent neural network (RNN), processes sequential data in a way that keeps track of the order as the data is input. RNNs maintain a hidden state or memory of past inputs thanks to connections that loop back on themselves. Natural language processing, time

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

series analysis, speech recognition, and other tasks are well suited for RNNs because of their recurrent structure. In an RNN, input data is composed of a sequence of elements, each representing a time step. In RNNs, a hidden state vector is updated at every step of the learning process. Memory for the network is encoded in this hidden state, which contains information about the previous inputs. RNNs are distinguished by their recurrent connections, which make it possible to influence the computation of the current time step with the hidden state from the previous time step. Using an update equation, an RNN calculates the new hidden state based on the current input and the previous hidden state. The new hidden state captures information from the current input and previous time steps. RNNs can produce outputs at every time step or just at the end of the process, depending on the task at hand. Sequential data can be captured by RNNs in terms of long-term dependencies. As a result of storing the entire sequence in the hidden state, the network can make predictions based on inputs from distant pasts. It is possible to generate text using RNNs, recognize speech, translate in a machine translator, analyse sentiment, and predict time series using RNNs. There are limitations to RNNs, such as the inability to capture as long-term dynamics as possible and the difficulty of maintaining training stability over time. Many other domains where sequential data analysis is essential have benefited from the use of Recurrent Neural Networks, which were pioneered in the field of natural language processing. Sequential patterns in data can be modelled and understood with their help.

4. Perceptron :(χ_4)
   As the foundation for more complex neural network architectures, perceptrons are among the simplest forms of neural networks. Data can be classified into two classes using the Perceptron based on input features, with the Perceptron mainly used for binary classification tasks. Inputs are represented by input neurons and outputs by the Perceptron or artificial neuron, which consists of a single layer of input nodes. A weight is assigned to each input node, representing the strength of the connection between that input and the Perceptron. Activation functions produce the output of the Perceptron by summing the inputs and applying a weighted sum. During training, the model learns parameters called perceptron. A Perceptron's decision is influenced by the weights of each input feature. By shifting the decision boundary of the Perceptron, the bias term affects decision making. As a general rule, the perceptron is used to separate data into binary classes, typically labelled as 0 and 1. Only linearly separable data can be classified by the Perceptron. A limitation of perceptron is that it cannot handle nonlinearly separable data. Unlike modern deep neural networks, Perceptrons are single-layer neural networks that can't capture hierarchical data representations. Artificial neural networks and machine learning were developed in part due to the Perceptron despite its limited capabilities. To address the limitations of the Perceptron and handle more complex tasks, such as deep learning, more advanced neural network architectures have been developed, such as multilayer perceptron (MLP) and deep neural network (DNN).

5. Gated recurrent unit (GRU):(χ_5)
   Recurrent neural networks (RNNs) can be modelled and learned from sequential data using gated recurrent units (GRUs), which reduce some of the computational complexity associated with LSTMs. Natural language processing, speech recognition, and time series analysis are among the tasks that GRUs are ideally designed to handle. Data is entered into GRUs as a sequence of time steps, each representing a different element in the sequence. Information is carried across time steps by a hidden state of GRUs. Information can be maintained and propagated through the network due to this hidden state. The information flow through the network is controlled by two update gates. Calculating the candidate activation depends on what information from the previous hidden state should be reset or ignored. The update gate incorporates new candidate activations into the new hidden state. GRUs calculate a candidate activation at each time step based on the current input and the reset gate. An estimation of the hidden state is provided by this candidate activation. The final hidden state is determined based on the candidate activation and the previous hidden state. While GRUs achieve similar results, they simplify some of the complexity of LSTM networks. They can sometimes be more computationally efficient because they have fewer gates and computations. Machine translation, sentiment analysis, speech recognition, and time series forecasting are just a few of the applications where GRUs are used. Data in sequential sequences can be captured and maintained using GRUs. In traditional RNNs, vanishing gradients are a common problem. GRUs are an effective and computationally efficient solution for sequence-related tasks, particularly when LSTMs are not required. For sequential data analysis, they are valuable tools because they can model sequential dependencies and learn from context.

6. Autoencoder:(χ_6)
   In unsupervised learning and dimensionality reduction, autoencoders are artificial neural networks. In

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

its primary function, it aims to minimize reconstruction errors in a lower-dimensional space by learning efficient data representations. Encoders and decoders work together to encrypt data into compact representations and then decode them to reconstruct the original data. Data compression and feature learning can be accomplished using autoencoders. This network translates input data into a lower dimensional representation that is called the "encoding" or "latent space". To reduce the dimensionality of the input data, the encoder employs linear transformations and activation functions applied to one or more layers of neurons. The encoded representation of the input data is produced in the final layer of the encoder. A compressed representation of input data is called an encoded representation. In autoencoders, latent spaces are used to capture the important features while discarding redundant information. It reconstructs the original input data from the encoded representation. Decoders and encoders are similar in that they both consist of layers of neurons, but they usually have a symmetrical structure. Decoding produces the reconstructed data in the final layer. In order to minimize the loss function measuring the difference between input and reconstructed data, autoencoders have to be trained to minimize a loss function. Binary cross entropy and mean squared error are two common loss functions. Data is encoded by learning a compressed representation.

7. Generative adversarial network (GAN) :(χ_7)
   In generative modeling, GANs are artificial neural networks that generate adversarial networks. In addition to generating images, videos, and texts, GANs can be used to generate a variety of data samples. Several neural networks are trained adversarial together to form a GAN, including the generator and discriminator. In addition to generating realistic images and videos, GANs can also generate text. Generators are neural networks that produce data samples from random noise or random vectors. An indistinguishable sample of data is the goal. Data samples with selected characteristics are generated through layers of the generator network. In addition to the discriminator, there is also the binary classifier, which is a neural network. In order to distinguish between the two types of data, the algorithm uses both real data samples and generated samples as input. Ultimately, it aims to learn how to distinguish real data samples from fake ones. Layers of the discriminator network are typically responsible for processing input data and generating a single output indicating whether the input is real. Generators are optimized so that their generated samples are classified as fake by the discriminator as few as possible. As a result, generator losses are minimized. Real samples are differentiated from fake samples by optimizing the discriminator. Losses from discriminators are minimized. Mode collapse occurs when the generator stops generating samples or produces only one mode of distribution of the data during GAN training. Images can be generated using GANs, styles can be transferred, images can be translated, data can be augmented, and super-resolution is possible. Computer graphics and art also use them to generate realistic content. A major contribution of generative adversarial networks has been the creation of highly realistic and diverse synthetic data with generative adversarial networks. A wide variety of applications are being developed in this area of research.

8. Recursive neural network (RecNN) :(χ_8)
   Trees or graphs are natural data structures on which recursive neural networks (RecNN) operate. RecNNs recursively apply the same neural network operation at different levels of the hierarchy, which allows them to handle variable-sized and structured data, while traditional neural networks can only handle fixed-sized inputs. Using these networks is particularly useful when processing hierarchical data representations, parsing natural language, and analysing tree-structured data. Syntactic parse trees, for example, or organizational hierarchies, are recursive data structures designed for recursive neural networks. A RecNN is a neural network that applies the same operation recursively at various levels. In addition to feedforward neural networks, convolutional layers are also suitable. An embedding represents the features or information associated with each node in the hierarchical structure. Combining their children's information and the recursive operation, embedding is calculated at each level. Children nodes pass information to parents in recursive neural networks. Parent nodes are represented through aggregation mechanisms through their child nodes. Recurrent neural networks can aggregate information using a variety of operations, including aggregation, averaging, and more complex functions. RecNNs typically produce the root node representation of a hierarchical structure as their final output. An entire hierarchy of features and information is encoded within this root representation. Natural language processing and bioinformatics use RecNNs for performing syntactic and semantic analysis, as well as predicting protein structures and analysing molecular structures. Recursive Neural Networks are an essential tool when dealing with structured or hierarchical data. In tasks like natural language parsing and understanding, where hierarchical data is prevalent, they allow for the extraction of meaningful representations from complex data structures.

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

9. Sequence to Sequence (Seq to Seq) model:(χ_9)
   A sequence-to-sequence model (Seq to Seq) is an architecture based on neural networks that is commonly used in natural language processing (NLP) and sequence generation tasks, such as speech recognition, machine translation and word summarization. Encoders convert variable-length input sequences into fixed-length context vectors known as latent representations or thought vectors. An input sequence's context vector captures important information about it and represents it densely. By summarizing the input sequence, the encoder produces a context vector. In other words, it encodes the meaning of the input in a fixed-size representation. Decoders take context vectors produced by encoders and synthesize variable-length output sequences. Each step of the output is determined by the context vector. As its parameters are adjusted, the model minimizes a loss function. There have been a number of ways that Seq to Seq models have been improved, including the use of attention mechanisms, beam searches, and more sophisticated architectures, such as the transformer, which is especially popular with its parallelization capabilities and outstanding NLP performance.

10. Self organizing map (SOM) :(χ_{10})
    This unsupervised machine learning method reduces dimensionality, clusters, and visualizes high dimensional data in a lower-dimensional space using a SOM, also known as a Kohonen map or a self-organizing feature map. This type of tool is generally useful for visualizing and recognizing patterns in data. It is particularly useful for exploring and organizing complex datasets. Two dimensional lattices of nodes form a SOM. The output space is often visualized as a map, with each neuron representing a point in it. According to the desired output dimensions, the user can choose a rectangular or hexagonal lattice. As the input data dimensionality increases, so does the weight vector associated with each neuron in the SOM. A small value or a random value is used to initialize these weight vectors. To improve the similarity between the winning neuron and its neighboring neurons, their weights are updated. In training, learning rate and neighborhood size parameters decrease with time, determining the extent of similarity among neurons and their influence on neighboring neurons. SOMs adjust to input data distribution through this weight update process. One of the most important features of SOMs is that they preserve topological relations between input and output data. The input space tends to map to the output space if the data points are close together. Visualizations of the data can be made possible thanks to this topological preservation. When data points are associated with their BMUs, the SOM can be used as a clustering technique. As an additional benefit, the SOM can be used to visualize data by mapping high-dimensional input data to the SOM grid of lower dimensions. Data exploration, pattern recognition, image analysis, segmenting customers, and extracting features are examples of fields in which SOMs can be used. A complex dataset can be identified by them if there are patterns, anomalies, or relationships. Visually intuitive Self-Organizing Maps make it easier to explore and understand high-dimensional data. Due to their ability to reveal hidden patterns and groupings through unsupervised learning, they are particularly useful when the data's structure and relationships are unclear in advance. Machine learning and artificial intelligence use a wide range of specialized architectures and variations designed to meet specific needs and challenges. These are just some of the major types of neural networks.

We want to select the best option from a large number of options. I have provided the following DM information:

1. Sequential data:
   The order of elements in sequential data matters. An RNN is a neural network that is designed to handle sequential data. By looping back on themselves, they are able to maintain hidden states or memories of past inputs. In tasks such as natural language processing or time series forecasting, RNNs are especially well-suited to the order of words in a sentence. Long-term dependencies in data can be challenging to capture with traditional RNNs due to the vanishing gradient problem. These issues were addressed by LSTMs and GRUs. Gating mechanisms are incorporated into LSTMs to control the information flow within the network. Speech recognition, machine translation, and sentiment analysis are among the applications where LSTMs have proven to be effective. LSTMs use gating mechanisms, but GRUs has fewer parameters and a simpler architecture. They are highly parallelizable and efficient since they rely on self-attention mechanisms. By treating the data as one-dimensional sequences, CNNs can also be adapted for sequential data. A TCN is a type of neural network designed to capture long-range dependency in sequential data. To learn from a broader context of the input sequence, they use dilated convolutions to increase the receptive field. Some tasks can be executed using hybrid models,

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

which combine elements from different architectures. Different neural network architectures can be used for sequential data based on the task at hand, the nature of the data, and the trade-offs between model complexity and performance. A variety of architectures are often explored by researchers and practitioners in order to find the one that is best suited to their needs.

2. Training:
In order to build and deploy machine learning models, neural networks must first be trained. Despite the differences in training techniques and algorithms between neural networks, the principles of training are generally the same. Backpropagation is used to train FNNs. Using the chain rule of calculus, losses are calculated with respect to each parameter in the network (weights and biases). In addition to FNNs, CNNs are trained using convolutional layers that are adapted from FNNs. Adding weights and operations to CNNs through convolution and pooling increases their complexity. These layers propagate gradients backwards. It creates a sequence of feedforward steps as it unrolls in time. Input sequence elements are represented by time steps. A gradient is computed at every time step and propagated backwards. There are problems associated with vanishing gradients when training RNNs. By utilizing LSTMs or GRUs, this issue can be mitigated. In LSTMs and GRUs, gates are specialized mechanisms similar to those used in traditional RNNs. Fake data are produced with Generator Training in order to simulate real data. Backpropagation of the discriminator's error results in gradients for the generator. Real data is differentiated from fake data with discriminator training. In order to create the gradients for the discriminator, it is necessary to back-propagate the error between its predictions and the actual labels. Unsupervised learning is typically used to train SOMs. The winning neuron in the SOM adjusts its weights to become more similar to its input data in response to input data. The weight space is smoothed out by neighbouring neurons updating their weights on a lesser scale. A predefined loss function is minimized by choosing optimal weights and biases during training. Depending on the neural network architecture and the nature of the task, the loss function, optimization algorithm, and hyperparameters can be selected. The best training results are often achieved through experimentation and tuning.

3. Object detection:
Computer vision tasks such as object detection involve identifying and locating objects within images or videos. Object detection tasks can be accomplished using various neural network architectures and techniques. In the early days of deep learning, R-CNN was one of the earliest methods used to detect objects. In this process, regions are proposed, features are extracted, and objects are classified. The region proposal is usually done with selective search, and the feature extraction is usually done with CNN. It is difficult to compute and slow to run R-CNNs, although they perform well in terms of accuracy. R-CNN can be improved by using one forward pass through the CNN and sharing computations for overlapping region proposals. Real-time YOLO (You Only Look Once) is a regression system that tackles the problem of object detection in real-time. This algorithm predicts bounding boxes for input images, class probabilities, and objectness scores directly using a grid. As compared to other methods, YOLO is known for its speed but may not have as good accuracy as Retina Net, a single-stage object detection method that incorporates a focal loss to account for imbalanced data. Two-stage detectors are fast and accurate at the same time. In addition to bounding boxes and class labels, CNN predicts pixel-level masks for objects using Faster R-CNN. In general, it is used to segment objects precisely. Scaling up network architecture enables Efficient detection to achieve a good balance between accuracy and computational efficiency. In order to achieve optimal performance across a variety of resource constraints, it uses a compound scaling method. Based on the detection of center points in objects and the regression of bounding boxes and key points directly from the centers, centernet is a single-stage object detection technique. In terms of accuracy and speed, it simplifies the object detection pipeline. An object detection neural network may use one or more of these architectures. A number of factors must be considered when choosing architecture, including accuracy, speed and computational resources. Models for greater accuracy and efficiency are being developed in the area of object detection, which continues to be a vibrant area of research.

4. Speech recognition:
Text is generated from spoken language through speech recognition. Voice assistants, transcription services, among others, use it as a crucial component. It is widely used to use RNNs, particularly LSTMs and Gated Recurrent Units (GRUs). Speech signals are characterized by temporal dependencies, which can be modelled using RNNs. However, CNNs can also be used for tasks other than image analysis. MFCC or spectrogram representations of audio are used for feature extraction. In addition to deep CNNs, Listen, Attend, and Spell (LAS) uses an encoder to decoder architecture along with

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

attention mechanisms to capture local patterns in acoustic signals. Input audio signals are processed by the encoder, while text is generated by the decoder. A wide range of tasks have been successfully performed by LAS models. An application's specific requirements and data size determine the neural network architecture to use for speech recognition. The simplicity and ability to train on a large amount of data have made end-to-end models more popular in recent years.

5. Time series analysis:
An analysis of a series of data points collected over time involves taking into account the patterns between them. A neural network can be used to forecast, detect anomalies, and classify time series effectively. Due to their ability to capture sequential dependencies, RNNs are well suited to time series data. RNNs process one data point per time step, and they store previous time step information in a hidden state. A longer sequence of data makes LSTMs and GRUs particularly effective for addressing the vanishing gradient problem. In tasks where inputs and outputs are sequences of varying lengths, Seq to Seq models, often based on RNNs, are often used. Speech recognition, machine translation, and time series forecasting are examples of applications of these models. Time series data can also be analysed by CNNs, which are primarily used for image data. For anomaly detection and classification tasks in time series analysis, they are used to extract local patterns and features from sequential data. Time series data can be processed using GANs. In order to capture long-range dependencies in the data, they use dilated convolutions that increase the receptive field. Forecasting and classification of time series have been successful using their method. Specifically designed for forecasting time series, autoregressive models use neural networks. Time series data with long-range dependencies are particularly well suited to LSTMs, a variant of RNNs. Data sets with multiple variables and changes over time can be handled by them. There are fewer parameters in GRUs than LSTMs, making them similar to LSTMs. Modelling time series data is also possible using them, especially when dealing with shorter sequences or when there is a limited amount of computing capacity available. The time series analysis of time series has been made possible by transformers, which were originally designed for natural language processing. Various time series tasks can be accomplished using transformers, owing to their self-attention mechanisms. It is important to consider the specific task, the properties of the data, and the resources available before choosing a neutral network architecture for time series analysis. In order to find the best model for a given application, researchers often experiment with different models.

Suppose that ten types of neutral networks (alternatives) such as $y = \{\chi_1, \chi_2, \chi_3, \chi_4, \chi_5, \chi_6, \chi_7, \chi_8, \chi_9, \chi_{10}\}$.
Step-1: Suppose that $[\mathcal{D}^l, \mathcal{D}^u] = \{[\mathcal{D}_i^l, \mathcal{D}_i^u] | i = 1, 2, 3, 4, 5\}$ is a finite set of decision makers, $y = \{\chi_i | i = 1, 2, ..., 10\}$ is the collection of diagnostic health imaging/alternatives and $D = \{e_i | i = 1, 2, ..., 5\}$ is a finite set family of parameters, where $e_1 =$ Sequential data $e_2 =$ Training, $e_3 =$ Object detection, $e_4 =$ Speech recognition, $e_5 =$ Time series analysis.
Step-2: Linguistic variables in the form of Very Good Report Presentation (VGRP) $= [0.9, 0.95]$, Good Report Presentation (GRP) $= [0.8, 0.9]$, Average Report Presentation (ARP) $= [0.65, 0.8]$, Poor Report Presentation (PRP) $= [0.5, 0.65]$, Very Poor Report Presentation (VPRP) $= [0.35, 0.5]$.
Form the weighted parameter matrix is given as

$$\mathcal{P}_1 = [\varpi_{ij}^l, \varpi_{ij}^u]_{5\times 5} = \begin{bmatrix} VGRP & VPRP & GRP & VPRP & ARP \\ GRP & VGRP & ARP & GRP & PRP \\ VPRP & ARP & VGRP & PRP & GRP \\ VPRP & GRP & VGRP & ARP & PRP \\ ARP & GRP & VPRP & VGRP & VPRP \end{bmatrix}$$

$$\mathcal{P}_2 = [\varpi_{ij}^l, \varpi_{ij}^u]_{5\times 5} = \begin{bmatrix} GRP & ARP & VPRP & VGRP & PRP \\ VPRP & GRP & VGRP & ARP & PRP \\ VPRP & PRP & GRP & VPRP & GRP \\ VGRP & VPRP & ARP & PRP & GRP \\ PRP & VGRP & ARP & GRP & VPRP \end{bmatrix}$$

$$\mathcal{P}_3 = [\varpi_{ij}^l, \varpi_{ij}^u]_{5\times 5} = \begin{bmatrix} ARP & GRP & PRP & VPRP & VGRP \\ PRP & ARP & VGRP & GRP & PRP \\ GRP & VGRP & GRP & PRP & ARP \\ VGRP & PRP & ARP & VPRP & GRP \\ GRP & VGRP & PRP & GRP & ARP \end{bmatrix}$$

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

$$\mathcal{P}_4 = [\varpi_{ij}^l, \varpi_{ij}^u]_{5\times 5} = \begin{bmatrix} PRP & VGRP & VGRP & ARP & VPRP \\ ARP & GRP & VPRP & VGRP & VPRP \\ VGRP & ARRP & VGRP & PRP & GRP \\ VGRP & VPRP & GRP & VPRP & ARP \\ GRP & VGRP & ARP & VPRP & GRP \end{bmatrix}$$

$$\mathcal{P}_5 = [\varpi_{ij}^l, \varpi_{ij}^u]_{5\times 5} = \begin{bmatrix} PRP & VGRP & ARP & GRP & PRP \\ ARP & GRP & PRP & VARP & GRP \\ VGRP & VPRP & GRP & ARP & VPRD \\ GRP & ARP & VPRP & VGRP & PRP \\ GRP & PRP & VGRP & VPRP & ARP \end{bmatrix}$$

$$\mathcal{P} = [\varpi_{ij}^l, \varpi_{ij}^u]_{5\times 5} = \begin{bmatrix} [0.64, 0.76] & [0.72, 0.82] & [0.64, 0.76] & [0.61, 0.73] & [0.58, 0.71] \\ [0.59, 0.73] & [0.79, 0.89] & [0.66, 0.77] & [0.7, 0.81] & [0.5, 0.64] \\ [0.61, 0.73] & [0.5, 0.65] & [0.84, 0.92] & [0.5, 0.65] & [0.68, 0.8] \\ [0.77, 0.85] & [0.53, 0.67] & [0.67, 0.79] & [0.55, 0.68] & [0.65, 0.78] \\ [0.71, 0.83] & [0.8, 0.88] & [0.61, 0.74] & [0.64, 0.75] & [0.56, 0.7] \end{bmatrix}$$

where $[\varpi_{ij}^l, \varpi_{ij}^u]$ be the weight provided by the specialist $[\mathcal{D}_i^l, \mathcal{D}_i^u]$ to each parameter e_j.
Step-3: Form the normalized weighted decision matrix is fallows

$$\widehat{\mathcal{N}} = \left[\widehat{n}_{ij}^l, \widehat{n}_{ij}^u\right]_{5\times 5} = \begin{bmatrix} [0.1641, 0.2289] & [0.1841, 0.2455] & [0.1608, 0.2222] & [0.1685, 0.2433] & [0.1598, 0.2391] \\ [0.1513, 0.2199] & [0.202, 0.2665] & [0.1658, 0.2251] & [0.1934, 0.27] & [0.1377, 0.2155] \\ [0.1564, 0.2199] & [0.1279, 0.1946] & [0.2111, 0.269] & [0.1381, 0.2167] & [0.1873, 0.2694] \\ [0.1974, 0.256] & [0.1355, 0.2006] & [0.1683, 0.231] & [0.1519, 0.2267] & [0.1791, 0.2626] \\ [0.1821, 0.25] & [0.2046, 0.2635] & [0.1533, 0.2164] & [0.1768, 0.25] & [0.1543, 0.2357] \end{bmatrix}.$$

weighted vector
$\mathcal{W} = ([0.0437, 0.0708], [0.0437, 0.0701], [0.0432, 0.0681], [0.0458, 0.0804], [0.0451, 0.0823])$.
Step-4: The aggregated decision matrix $\left[\mathcal{X}^l, \mathcal{X}^u\right] = \frac{\left[\mathcal{D}_1^l, \mathcal{D}_1^u\right] + \left[\mathcal{D}_2^l, \mathcal{D}_2^u\right] + \ldots + \left[\mathcal{D}_5^l, \mathcal{D}_5^u\right]}{5}$

$$\mathcal{D}_1 = \begin{bmatrix} ([0.52, 0.7], [0.3, 0.55], [0.3, 0.48]) & ([0.45, 0.5], [0.5, 0.55], [0.5, 0.55]) & ([0.6, 0.65], [0.45, 0.5], [0.35, 0.4]) \\ ([0.43, 0.55], [0.4, 0.5], [0.45, 0.57]) & ([0.5, 0.55], [0.6, 0.65], [0.45, 0.5]) & ([0.65, 0.7], [0.35, 0.55], [0.3, 0.35]) \\ ([0.53, 0.6], [0.5, 0.65], [0.4, 0.47]) & ([0.45, 0.5], [0.4, 0.5], [0.5, 0.55]) & ([0.5, 0.55], [0.3, 0.4], [0.45, 0.5]) \\ ([0.48, 0.5], [0.35, 0.45], [0.5, 0.52]) & ([0.4, 0.55], [0.3, 0.45], [0.45, 0.6]) & ([0.7, 0.75], [0.55, 0.6], [0.25, 0.3]) \\ ([0.53, 0.6], [0.45, 0.5], [0.4, 0.47]) & ([0.35, 0.45], [0.4, 0.45], [0.55, 0.65]) & ([0.45, 0.5], [0.6, 0.65], [0.5, 0.55]) \\ ([0.6, 0.8], [0.65, 0.7], [0.2, 0.4]) & ([0.4, 0.5], [0.25, 0.3], [0.5, 0.6]) & ([0.35, 0.45], [0.4, 0.45], [0.55, 0.65]) \\ ([0.3, 0.5], [0.5, 0.6], [0.5, 0.7]) & ([0.35, 0.4], [0.35, 0.4], [0.6, 0.65]) & ([0.5, 0.55], [0.35, 0.4], [0.45, 0.5]) \\ ([0.3, 0.4], [0.55, 0.65], [0.6, 0.7]) & ([0.45, 0.55], [0.4, 0.5], [0.45, 0.55]) & ([0.25, 0.3], [0.45, 0.55], [0.7, 0.75]) \\ ([0.45, 0.6], [0.45, 0.55], [0.4, 0.55]) & ([0.4, 0.55], [0.5, 0.55], [0.45, 0.6]) & ([0.4, 0.45], [0.55, 0.6], [0.55, 0.6]) \\ ([0.35, 0.5], [0.65, 0.7], [0.5, 0.65]) & ([0.5, 0.6], [0.6, 0.65], [0.4, 0.5]) & ([0.5, 0.55], [0.6, 0.65], [0.45, 0.5]) \\ ([0.5, 0.55], [0.5, 0.6], [0.45, 0.5]) & ([0.35, 0.45], [0.45, 0.5], [0.55, 0.65]) & \\ ([0.55, 0.6], [0.7, 0.75], [0.4, 0.45]) & ([0.45, 0.5], [0.25, 0.35], [0.5, 0.55]) & \\ ([0.45, 0.5], [0.25, 0.45], [0.5, 0.55]) & ([0.25, 0.3], [0.2, 0.5], [0.7, 0.75]) & \\ ([0.65, 0.7], [0.4, 0.5], [0.3, 0.35]) & ([0.3, 0.35], [0.35, 0.5], [0.65, 0.7]) & \\ ([0.45, 0.55], [0.5, 0.6], [0.45, 0.55]) & ([0.45, 0.5], [0.45, 0.65], [0.5, 0.55]) & \\ ([0.35, 0.4], [0.45, 0.5], [0.6, 0.65]) & ([0.3, 0.35], [0.35, 0.4], [0.65, 0.7]) & \\ ([0.25, 0.35], [0.5, 0.55], [0.65, 0.75]) & ([0.25, 0.3], [0.5, 0.55], [0.7, 0.75]) & \\ ([0.4, 0.45], [0.55, 0.6], [0.55, 0.6]) & ([0.3, 0.4], [0.55, 0.7], [0.6, 0.7]) & \\ ([0.5, 0.55], [0.35, 0.45], [0.45, 0.5]) & ([0.35, 0.4], [0.6, 0.7], [0.6, 0.65]) & \\ ([0.6, 0.65], [0.25, 0.35], [0.35, 0.4]) & ([0.45, 0.55], [0.5, 0.6], [0.45, 0.55]) & \end{bmatrix}$$

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

$$\mathcal{D}_2 = \begin{bmatrix} ([0.5,0.7],[0.45,0.6],[0.3,0.5]) & ([0.4,0.45],[0.65,0.7],[0.55,0.6]) & ([0.45,0.5],[0.35,0.45],[0.5,0.55]) \\ ([0.4,0.45],[0.55,0.65],[0.55,0.6]) & ([0.65,0.7],[0.5,0.55],[0.3,0.35]) & ([0.6,0.7],[0.2,0.25],[0.3,0.4]) \\ ([0.5,0.55],[0.6,0.7],[0.45,0.5]) & ([0.55,0.6],[0.45,0.5],[0.4,0.45]) & ([0.5,0.55],[0.5,0.55],[0.45,0.5]) \\ ([0.45,0.6],[0.4,0.55],[0.4,0.55]) & ([0.4,0.55],[0.3,0.35],[0.45,0.6]) & ([0.45,0.55],[0.55,0.65],[0.45,0.55]) \\ ([0.5,0.55],[0.45,0.55],[0.45,0.5]) & ([0.35,0.5],[0.4,0.5],[0.5,0.65]) & ([0.35,0.45],[0.45,0.6],[0.55,0.65]) \\ ([0.6,0.65],[0.65,0.7],[0.35,0.4]) & ([0.55,0.65],[0.55,0.6],[0.35,0.45]) & ([0.3,0.4],[0.65,0.7],[0.6,0.7]) \\ ([0.3,0.6],[0.4,0.6],[0.4,0.7]) & ([0.45,0.55],[0.45,0.5],[0.45,0.55]) & ([0.5,0.55],[0.6,0.65],[0.45,0.5]) \\ ([0.65,0.7],[0.6,0.7],[0.3,0.35]) & ([0.3,0.45],[0.55,0.6],[0.55,0.7]) & ([0.25,0.3],[0.45,0.5],[0.7,0.75]) \\ ([0.55,0.6],[0.5,0.65],[0.4,0.45]) & ([0.4,0.55],[0.35,0.5],[0.45,0.6]) & ([0.4,0.45],[0.5,0.55],[0.55,0.6]) \\ ([0.3,0.4],[0.65,0.7],[0.6,0.7]) & ([0.5,0.6],[0.4,0.45],[0.4,0.5]) & ([0.3,0.35],[0.35,0.45],[0.65,0.7]) \\ \\ ([0.6,0.65],[0.65,0.7],[0.35,0.4]) & ([0.35,0.4],[0.45,0.55],[0.6,0.65]) & \\ ([0.7,0.75],[0.45,0.55],[0.25,0.3]) & ([0.45,0.5],[0.2,0.35],[0.5,0.55]) & \\ ([0.45,0.5],[0.35,0.45],[0.5,0.55]) & ([0.5,0.55],[0.25,0.3],[0.45,0.5]) & \\ ([0.5,0.55],[0.4,0.5],[0.45,0.5]) & ([0.3,0.35],[0.35,0.5],[0.65,0.7]) & \\ ([0.45,0.65],[0.5,0.6],[0.35,0.55]) & ([0.5,0.55],[0.45,0.65],[0.45,0.5]) & \\ ([0.35,0.45],[0.35,0.5],[0.55,0.65]) & ([0.45,0.5],[0.3,0.4],[0.5,0.55]) & \\ ([0.45,0.5],[0.4,0.45],[0.5,0.55]) & ([0.55,0.65],[0.35,0.4],[0.35,0.45]) & \\ ([0.5,0.65],[0.5,0.55],[0.35,0.5]) & ([0.35,0.4],[0.2,0.25],[0.6,0.65]) & \\ ([0.4,0.45],[0.35,0.55],[0.55,0.6]) & ([0.2,0.4],[0.25,0.5],[0.6,0.8]) & \\ ([0.25,0.4],[0.5,0.65],[0.6,0.75]) & ([0.45,0.5],[0.5,0.6],[0.5,0.55]) & \end{bmatrix}$$

$$\mathcal{D}_3 = \begin{bmatrix} ([0.6,0.8],[0.45,0.6],[0.2,0.4]) & ([0.6,0.65],[0.45,0.6],[0.35,0.4]) & ([0.4,0.45],[0.5,0.6],[0.55,0.6]) \\ ([0.5,0.55],[0.55,0.65],[0.45,0.5]) & ([0.55,0.6],[0.6,0.65],[0.4,0.45]) & ([0.5,0.55],[0.55,0.65],[0.45,0.5]) \\ ([0.55,0.6],[0.65,0.7],[0.4,0.45]) & [0.45,0.5],[0.5,0.6],[0.5,0.55]) & ([0.5,0.55],[0.25,0.3],[0.45,0.5]) \\ ([0.45,0.55],[0.45,0.55],[0.45,0.55]) & ([0.4,0.5],[0.45,0.55],[0.5,0.6]) & ([0.6,0.65],[0.45,0.5],[0.35,0.4]) \\ ([0.5,0.6],[0.5,0.6],[0.4,0.5]) & ([0.25,0.3],[0.4,0.45],[0.7,0.75]) & ([0.5,0.55],[0.4,0.45],[0.45,0.5]) \\ ([0.6,0.65],[0.6,0.7],[0.35,0.4]) & ([0.35,0.5],[0.35,0.4],[0.5,0.65]) & ([0.35,0.45],[0.5,0.55],[0.55,0.65]) \\ ([0.3,0.5],[0.2,0.4],[0.5,0.7]) & ([0.55,0.6],[0.25,0.3],[0.4,0.45]) & ([0.55,0.65],[0.25,0.35],[0.35,0.45]) \\ ([0.4,0.45],[0.4,0.45],[0.55,0.6]) & ([0.5,0.55],[0.5,0.6],[0.45,0.5]) & ([0.25,0.3],[0.45,0.6],[0.7,0.75]) \\ ([0.5,0.55],[0.55,0.6],[0.45,0.5]) & ([0.65,0.7],[0.45,0.5],[0.3,0.35]) & ([0.5,0.55],[0.35,0.45],[0.45,0.5]) \\ ([0.25,0.35],[0.45,0.65],[0.65,0.75]) & ([0.45,0.5],[0.55,0.6],[0.5,0.55]) & ([0.55,0.6],[0.45,0.5],[0.4,0.45]) \\ ([0.6,0.65],[0.45,0.5],[0.35,0.4]) & ([0.45,0.5],[0.4,0.45],[0.5,0.55]) & \\ ([0.5,0.55],[0.5,0.6],[0.45,0.5]) & ([0.55,0.6],[0.45,0.5],[0.4,0.45]) & \\ ([0.35,0.45],[0.35,0.4],[0.55,0.65]) & ([0.5,0.55],[0.35,0.5],[0.45,0.5]) & \\ ([0.45,0.6],[0.4,0.45],[0.4,0.55]) & ([0.5,0.55],[0.35,0.45],[0.45,0.5]) & \\ ([0.5,0.55],[0.5,0.55],[0.45,0.5]) & ([0.3,0.45],[0.45,0.5],[0.55,0.7]) & \\ ([0.45,0.6],[0.55,0.6],[0.4,0.55]) & ([0.45,0.5],[0.3,0.4],[0.5,0.55]) & \\ ([0.35,0.45],[0.45,0.7],[0.55,0.65]) & ([0.25,0.3],[0.5,0.55],[0.7,0.75]) & \\ ([0.5,0.55],[0.5,0.55],[0.45,0.5]) & ([0.3,0.35],[0.6,0.7],[0.65,0.7]) & \\ ([0.4,0.45],[0.4,0.45],[0.55,0.6]) & ([0.2,0.25],[0.6,0.75],[0.75,0.8]) & \\ ([0.35,0.5],[0.5,0.65],[0.5,0.65]) & ([0.3,0.35],[0.45,0.6],[0.65,0.7]) & \end{bmatrix}$$

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

$$\mathcal{D}_4 = \begin{bmatrix} ([0.35,0.6],[0.6,0.75],[0.4,0.65]) & ([0.7,0.75],[0.35,0.5],[0.25,0.3]) & ([0.65,0.7],[0.5,0.65],[0.3,0.35]) \\ ([0.4,0.5],[0.65,0.7],[0.5,0.6]) & ([0.55,0.6],[0.25,0.4],[0.4,0.45]) & ([0.55,0.6],[0.55,0.6],[0.4,0.45]) \\ ([0.35,0.45],[0.55,0.6],[0.55,0.65]) & ([0.45,0.6],[0.45,0.5],[0.4,0.55]) & ([0.5,0.55],[0.25,0.35],[0.45,0.5]) \\ ([0.45,0.55],[0.45,0.65],[0.45,0.55]) & ([0.4,0.55],[0.3,0.35],[0.45,0.6]) & ([0.35,0.4],[0.45,0.55],[0.6,0.65]) \\ ([0.45,0.6],[0.65,0.7],[0.4,0.55]) & ([0.35,0.45],[0.5,0.55],[0.55,0.65]) & ([0.25,0.3],[0.55,0.6],[0.7,0.75]) \\ ([0.4,0.45],[0.35,0.4],[0.55,0.6]) & ([0.4,0.45],[0.25,0.3],[0.55,0.6]) & ([0.35,0.4],[0.5,0.65],[0.6,0.65]) \\ ([0.3,0.5],[0.2,0.35],[0.5,0.7]) & ([0.6,0.65],[0.7,0.75],[0.35,0.4]) & ([0.45,0.55],[0.25,0.35],[0.45,0.55]) \\ ([0.5,0.55],[0.45,0.5],[0.45,0.5]) & ([0.65,0.7],[0.45,0.6],[0.3,0.35]) & ([0.25,0.3],[0.55,0.7],[0.7,0.75]) \\ ([0.55,0.6],[0.35,0.4],[0.4,0.45]) & ([0.45,0.55],[0.55,0.65],[0.45,0.55]) & ([0.4,0.45],[0.35,0.45],[0.55,0.6]) \\ ([0.45,0.5],[0.45,0.6],[0.5,0.55]) & ([0.55,0.6],[0.35,0.45],[0.4,0.45]) & ([0.5,0.55],[0.6,0.65],[0.45,0.5]) \\ ([0.4,0.45],[0.6,0.7],[0.55,0.6]) & ([0.4,0.6],[0.25,0.3],[0.4,0.6]) & \\ ([0.55,0.6],[0.4,0.45],[0.4,0.45]) & ([0.3,0.6],[0.3,0.35],[0.4,0.7]) & \\ ([0.35,0.4],[0.5,0.6],[0.6,0.65]) & ([0.2,0.6],[0.35,0.5],[0.4,0.8]) & \\ ([0.45,0.5],[0.4,0.45],[0.5,0.55]) & ([0.3,0.5],[0.35,0.4],[0.5,0.7]) & \\ ([0.5,0.55],[0.55,0.6],[0.45,0.5]) & ([0.5,0.7],[0.45,0.6],[0.3,0.5]) & \\ ([0.6,0.65],[0.35,0.4],[0.35,0.4]) & ([0.45,0.49],[0.3,0.7],[0.51,0.55]) & \\ ([0.35,0.45],[0.45,0.5],[0.55,0.65]) & ([0.55,0.6],[0.5,0.55],[0.4,0.45]) & \\ ([0.45,0.5],[0.6,0.65],[0.5,0.55]) & ([0.35,0.45],[0.55,0.7],[0.55,0.65]) & \\ ([0.4,0.5],[0.35,0.45],[0.5,0.6]) & ([0.55,0.45],[0.6,0.65],[0.55,0.45]) & \\ ([0.35,0.55],[0.2,0.5],[0.45,0.65]) & ([0.6,0.5],[0.5,0.6],[0.5,0.4]) & \end{bmatrix}$$

$$\mathcal{D}_5 = \begin{bmatrix} ([0.45,0.6],[0.35,0.45],[0.4,0.55]) & ([0.65,0.7],[0.5,0.6],[0.3,0.35]) & ([0.4,0.45],[0.45,0.55],[0.55,0.6]) \\ ([0.4,0.5],[0.55,0.6],[0.5,0.6]) & ([0.4,0.45],[0.6,0.65],[0.55,0.6]) & ([0.55,0.6],[0.5,0.6],[0.4,0.45]) \\ ([0.5,0.55],[0.65,0.7],[0.45,0.5]) & ([0.5,0.55],[0.45,0.5],[0.45,0.5]) & ([0.4,0.45],[0.25,0.35],[0.55,0.6]) \\ ([0.35,0.45],[0.45,0.5],[0.55,0.65]) & ([0.4,0.5],[0.3,0.35],[0.5,0.6]) & ([0.35,0.4],[0.45,0.5],[0.6,0.65]) \\ ([0.45,0.5],[0.25,0.3],[0.5,0.55]) & ([0.35,0.4],[0.6,0.65],[0.6,0.65]) & ([0.45,0.5],[0.6,0.65],[0.5,0.55]) \\ ([0.6,0.65],[0.35,0.4],[0.35,0.4]) & ([0.45,0.5],[0.25,0.3],[0.5,0.55]) & ([0.35,0.4],[0.5,0.55],[0.6,0.65]) \\ ([0.3,0.6],[0.25,0.5],[0.4,0.7]) & ([0.35,0.45],[0.35,0.45],[0.55,0.65]) & ([0.5,0.55],[0.45,0.6],[0.45,0.5]) \\ ([0.45,0.5],[0.5,0.55],[0.5,0.55]) & ([0.45,0.5],[0.3,0.35],[0.5,0.55]) & ([0.25,0.3],[0.5,0.65],[0.7,0.75]) \\ ([0.4,0.55],[0.45,0.55],[0.45,0.6]) & ([0.6,0.65],[0.5,0.6],[0.35,0.4]) & ([0.4,0.45],[0.6,0.7],[0.55,0.6]) \\ ([0.3,0.35],[0.5,0.65],[0.65,0.7]) & ([0.55,0.6],[0.4,0.45],[0.4,0.45]) & ([0.3,0.35],[0.65,0.75],[0.65,0.7]) \\ ([0.6,0.65],[0.5,0.65],[0.35,0.4]) & ([0.55,0.6],[0.5,0.55],[0.4,0.45]) & \\ ([0.35,0.45],[0.65,0.7],[0.55,0.65]) & ([0.45,0.6],[0.4,0.45],[0.4,0.55]) & \\ ([0.4,0.5],[0.6,0.7],[0.5,0.6]) & ([0.5,0.6],[0.45,0.5],[0.4,0.5]) & \\ ([0.55,0.65],[0.35,0.55],[0.35,0.45]) & ([0.3,0.5],[0.35,0.55],[0.5,0.7]) & \\ ([0.45,0.55],[0.45,0.6],[0.45,0.55]) & ([0.35,0.7],[0.25,0.5],[0.3,0.65]) & \\ ([0.35,0.45],[0.6,0.65],[0.55,0.65]) & ([0.45,0.49],[0.3,0.4],[0.51,0.55]) & \\ ([0.6,0.65],[0.55,0.7],[0.35,0.4]) & ([0.55,0.6],[0.45,0.5],[0.4,0.45]) & \\ ([0.55,0.65],[0.5,0.6],[0.35,0.45]) & ([0.35,0.45],[0.35,0.4],[0.55,0.65]) & \\ ([0.4,0.45],[0.3,0.35],[0.55,0.6]) & ([0.5,0.45],[0.5,0.6],[0.55,0.5]) & \\ ([0.6,0.65],[0.6,0.65],[0.35,0.4]) & ([0.45,0.5],[0.25,0.3],[0.5,0.55]) & \end{bmatrix}$$

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

$$\mathcal{D} = \begin{bmatrix} ([0.48,0.68],[0.43,0.59],[0.32,0.52]) & ([0.56,0.61],[0.49,0.59],[0.39,0.44]) & ([0.5,0.55],[0.45,0.55],[0.45,0.5]) \\ ([0.43,0.51],[0.54,0.62],[0.49,0.57]) & ([0.53,0.58],[0.51,0.58],[0.42,0.47]) & ([0.57,0.63],[0.43,0.53],[0.37,0.43]) \\ ([0.49,0.55],[0.59,0.67],[0.45,0.51]) & ([0.48,0.55],[0.45,0.52],[0.45,0.52]) & ([0.48,0.53],[0.31,0.39],[0.47,0.52]) \\ ([0.44,0.53],[0.42,0.54],[0.47,0.56]) & ([0.4,0.53],[0.33,0.41],[0.47,0.6]) & ([0.49,0.55],[0.49,0.56],[0.45,0.51]) \\ ([0.49,0.57],[0.46,0.53],[0.43,0.51]) & ([0.33,0.42],[0.46,0.52],[0.58,0.67]) & ([0.4,0.46],[0.52,0.59],[0.54,0.6]) \\ ([0.56,0.64],[0.52,0.58],[0.36,0.44]) & ([0.43,0.52],[0.33,0.38],[0.48,0.57]) & ([0.34,0.42],[0.51,0.58],[0.58,0.66]) \\ ([0.3,0.54],[0.31,0.49],[0.46,0.7]) & ([0.46,0.53],[0.42,0.48],[0.47,0.54]) & ([0.5,0.57],[0.38,0.47],[0.43,0.5]) \\ ([0.46,0.52],[0.5,0.57],[0.48,0.54]) & ([0.47,0.55],[0.44,0.53],[0.45,0.53]) & ([0.25,0.3],[0.48,0.6],[0.7,0.75]) \\ ([0.49,0.58],[0.46,0.55],[0.42,0.51]) & ([0.5,0.6],[0.47,0.56],[0.4,0.5]) & ([0.42,0.47],[0.47,0.55],[0.53,0.58]) \\ ([0.33,0.42],[0.54,0.66],[0.58,0.67]) & ([0.51,0.58],[0.46,0.52],[0.42,0.49]) & ([0.43,0.48],[0.53,0.6],[0.52,0.57]) \\ ([0.54,0.59],[0.54,0.63],[0.41,0.46]) & ([0.42,0.51],[0.41,0.47],[0.49,0.58]) & \\ ([0.53,0.59],[0.54,0.61],[0.41,0.47]) & ([0.44,0.56],[0.32,0.4],[0.44,0.56]) & \\ ([0.4,0.47],[0.41,0.52],[0.53,0.6]) & ([0.39,0.52],[0.32,0.46],[0.48,0.61]) & \\ ([0.52,0.6],[0.39,0.49],[0.4,0.48]) & ([0.34,0.45],[0.35,0.48],[0.55,0.66]) & \\ ([0.47,0.57],[0.5,0.59],[0.43,0.53]) & ([0.42,0.58],[0.41,0.58],[0.42,0.58]) & \\ ([0.42,0.51],[0.46,0.53],[0.49,0.58]) & ([0.42,0.47],[0.31,0.46],[0.53,0.58]) & \\ ([0.4,0.48],[0.47,0.58],[0.52,0.6]) & ([0.43,0.49],[0.46,0.51],[0.51,0.57]) & \\ ([0.48,0.56],[0.53,0.59],[0.44,0.52]) & ([0.33,0.41],[0.45,0.55],[0.59,0.67]) & \\ ([0.42,0.48],[0.35,0.45],[0.52,0.58]) & ([0.36,0.39],[0.51,0.64],[0.61,0.64]) & \\ ([0.43,0.55],[0.41,0.56],[0.45,0.57]) & ([0.45,0.48],[0.44,0.54],[0.52,0.55]) & \end{bmatrix}$$

$$= \left[x_{jk}^{l}, x_{jk}^{u}\right]_{10\times 5}$$

Step-5: The weighted q-rung NVSS decision matrix $\left[\mathcal{Y}^{l}, \mathcal{Y}^{u}\right] = \left[\mu_{k}^{l} \times x_{jk}^{l}, \mu_{k}^{u} \times x_{jk}^{u}\right]$

$$= \begin{bmatrix} ([0.021,0.0481],[0.0188,0.0418],[0.014,0.0368]) & ([0.0245,0.0428],[0.0214,0.0414],[0.017,0.0308]) \\ ([0.0188,0.0361],[0.0236,0.0439],[0.0214,0.0403]) & ([0.0232,0.0407],[0.0223,0.0407],[0.0184,0.0329]) \\ ([0.0214,0.0389],[0.0258,0.0474],[0.0196,0.0361]) & ([0.021,0.0386],[0.0197,0.0365],[0.0197,0.0365]) \\ ([0.0192,0.0375],[0.0183,0.0382],[0.0205,0.0396]) & ([0.0175,0.0372],[0.0144,0.0287],[0.0205,0.0421]) \\ ([0.0214,0.0403],[0.0201,0.0375],[0.0188,0.0361]) & ([0.0144,0.0294],[0.0201,0.0365],[0.0253,0.047]) \\ ([0.0244,0.0453],[0.0227,0.041],[0.0157,0.0311]) & ([0.0188,0.0365],[0.0144,0.0266],[0.021,0.04]) \\ ([0.0131,0.0382],[0.0135,0.0347],[0.0201,0.0495]) & ([0.0201,0.0372],[0.0184,0.0336],[0.0205,0.0379]) \\ ([0.0201,0.0368],[0.0218,0.0403],[0.021,0.0382]) & ([0.0205,0.0386],[0.0192,0.0372],[0.0197,0.0372]) \\ ([0.0214,0.041],[0.0201,0.0389],[0.0183,0.0361]) & ([0.0218,0.0421],[0.0205,0.0393],[0.0175,0.035]) \\ ([0.0144,0.0297],[0.0236,0.0467],[0.0253,0.0474]) & ([0.0223,0.0407],[0.0201,0.0365],[0.0184,0.0343]) \\ \\ ([0.0216,0.0374],[0.0194,0.0374],[0.0194,0.034]) & ([0.0247,0.0475],[0.0247,0.0507],[0.0188,0.037]) \\ ([0.0246,0.0429],[0.0186,0.0361],[0.016,0.0293]) & ([0.0243,0.0475],[0.0247,0.0491],[0.0188,0.0378]) \\ ([0.0207,0.0361],[0.0134,0.0265],[0.0203,0.0354]) & ([0.0183,0.0378],[0.0188,0.0418],[0.0243,0.0483]) \\ ([0.0212,0.0374],[0.0212,0.0381],[0.0194,0.0347]) & ([0.0238,0.0483],[0.0179,0.0394],[0.0183,0.0386]) \\ ([0.0173,0.0313],[0.0225,0.0402],[0.0233,0.0408]) & ([0.0215,0.0459],[0.0229,0.0475],[0.0197,0.0426]) \\ ([0.0147,0.0286],[0.022,0.0395],[0.025,0.0449]) & ([0.0192,0.041],[0.0211,0.0426],[0.0224,0.0467]) \\ ([0.0216,0.0388],[0.0164,0.032],[0.0186,0.034]) & ([0.0183,0.0386],[0.0215,0.0467],[0.0238,0.0483]) \\ ([0.0108,0.0204],[0.0207,0.0408],[0.0302,0.051]) & ([0.022,0.045],[0.0243,0.0475],[0.0201,0.0418]) \\ ([0.0181,0.032],[0.0203,0.0374],[0.0229,0.0395]) & ([0.0192,0.0386],[0.016,0.0362],[0.0238,0.0467]) \\ ([0.0186,0.0327],[0.0229,0.0408],[0.0225,0.0388]) & ([0.0197,0.0442],[0.0188,0.045],[0.0206,0.0459]) \\ \\ ([0.0189,0.042],[0.0185,0.0387],[0.0221,0.0477]) & \\ ([0.0198,0.0461],[0.0144,0.0329],[0.0198,0.0461]) & \\ ([0.0176,0.0428],[0.0144,0.0379],[0.0216,0.0502]) & \\ ([0.0153,0.037],[0.0158,0.0395],[0.0248,0.0543]) & \\ ([0.0189,0.0477],[0.0185,0.0477],[0.0189,0.0477]) & \\ ([0.0189,0.0387],[0.014,0.0379],[0.0239,0.0477]) & \\ ([0.0194,0.0403],[0.0207,0.042],[0.023,0.0469]) & \\ ([0.0149,0.0337],[0.0203,0.0453],[0.0266,0.0551]) & \\ ([0.0162,0.0321],[0.023,0.0527],[0.0275,0.0527]) & \\ ([0.0203,0.0395],[0.0198,0.0444],[0.0234,0.0453]) & \end{bmatrix}$$

$$= \left[\chi_{jk}^{l}, \chi_{jk}^{u}\right]_{10\times 5}$$

Step-6: Determine the values for q-rung NVSS-PIS and q-rung NVSS-NIS. Now,

https://doi.org/10.54216/IJNS.220406
Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

$\left[\chi^{l+}, \chi^{u+}\right]$	q-rung NVSS-PIS
$\left[\chi_1^{l+}, \chi_1^{u+}\right]$	$([0.0247, 0.0481], [0.0185, 0.0374], [0.014, 0.0308])$
$\left[\chi_2^{l+}, \chi_2^{u+}\right]$	$([0.0246, 0.0475], [0.0144, 0.0329], [0.016, 0.0293])$
$\left[\chi_3^{l+}, \chi_3^{u+}\right]$	$([0.0214, 0.0428], [0.0134, 0.0265], [0.0196, 0.0354])$
$\left[\chi_4^{l+}, \chi_4^{u+}\right]$	$([0.0238, 0.0483], [0.0144, 0.0287], [0.0183, 0.0347])$
$\left[\chi_5^{l+}, \chi_5^{u+}\right]$	$([0.0215, 0.0477], [0.0185, 0.0365], [0.0188, 0.0361])$
$\left[\chi_6^{l+}, \chi_6^{u+}\right]$	$([0.0244, 0.0453], [0.014, 0.0266], [0.0157, 0.0311])$
$\left[\chi_7^{l+}, \chi_7^{u+}\right]$	$([0.0216, 0.0403], [0.0135, 0.032], [0.0186, 0.034])$
$\left[\chi_8^{l+}, \chi_8^{u+}\right]$	$([0.022, 0.045], [0.0192, 0.0372], [0.0197, 0.0372])$
$\left[\chi_9^{l+}, \chi_9^{u+}\right]$	$([0.0218, 0.0421], [0.016, 0.0362], [0.0175, 0.035])$
$\left[\chi_{10}^{l+}, \chi_{10}^{u+}\right]$	$([0.0223, 0.0442], [0.0188, 0.0365], [0.0184, 0.0343])$
$\left[\chi^{l-}, \chi^{u-}\right]$	q-rung NVSS-NIS
$\left[\chi_1^{l-}, \chi_1^{u-}\right]$	$([0.0189, 0.0374], [0.0247, 0.0507], [0.0221, 0.0477])$
$\left[\chi_2^{l-}, \chi_2^{u-}\right]$	$([0.0188, 0.0361], [0.0247, 0.0491], [0.0214, 0.0461])$
$\left[\chi_3^{l-}, \chi_3^{u-}\right]$	$([0.0176, 0.0361], [0.0258, 0.0474], [0.0243, 0.0502])$
$\left[\chi_4^{l-}, \chi_4^{u-}\right]$	$([0.0153, 0.037], [0.0212, 0.0395], [0.0248, 0.0543])$
$\left[\chi_5^{l-}, \chi_5^{u-}\right]$	$([0.0144, 0.0294], [0.0229, 0.0477], [0.0253, 0.0477])$
$\left[\chi_6^{l-}, \chi_6^{u-}\right]$	$([0.0147, 0.0286], [0.0227, 0.0426], [0.025, 0.0477])$
$\left[\chi_7^{l-}, \chi_7^{u-}\right]$	$([0.0131, 0.0372], [0.0215, 0.0467], [0.0238, 0.0495])$
$\left[\chi_8^{l-}, \chi_8^{u-}\right]$	$([0.0108, 0.0204], [0.0243, 0.0475], [0.0302, 0.0551])$
$\left[\chi_9^{l-}, \chi_9^{u-}\right]$	$([0.0162, 0.032], [0.023, 0.0527], [0.0275, 0.0527])$
$\left[\chi_{10}^{l-}, \chi_{10}^{u-}\right]$	$([0.0144, 0.0297], [0.0236, 0.0467], [0.0253, 0.0474])$

Step-7: The q-rung NVSS EDs from q-rung NVSS-PIS and q-rung NVSS-NIS.

$\left[(\chi_i^l), (\chi_i^u)\right]$	$\left[d_i^{l+}, d_i^{u+}\right]$	$\left[d_i^{l-}, d_i^{u-}\right]$
$\left[(\chi_1^l), (\chi_1^u)\right]$	[0.0153, 0.0276]	[0.0173, 0.0382]
$\left[(\chi_2^l), (\chi_2^u)\right]$	[0.0196, 0.0338]	[0.0168, 0.0372]
$\left[(\chi_3^l), (\chi_3^u)\right]$	[0.0165, 0.0372]	[0.0217, 0.0368]
$\left[(\chi_4^l), (\chi_4^u)\right]$	[0.0164, 0.037]	[0.0181, 0.0353]
$\left[(\chi_5^l), (\chi_5^u)\right]$	[0.0134, 0.0353]	[0.0168, 0.035]
$\left[(\chi_6^l), (\chi_6^u)\right]$	[0.0246, 0.0442]	[0.0203, 0.0351]
$\left[(\chi_7^l), (\chi_7^u)\right]$	[0.017, 0.0311]	[0.0185, 0.0308]
$\left[(\chi_8^l), (\chi_8^u)\right]$	[0.0194, 0.0398]	[0.0264, 0.0489]
$\left[(\chi_9^l), (\chi_9^u)\right]$	[0.0181, 0.0311]	[0.019, 0.0436]
$\left[(\chi_{10}^l), (\chi_{10}^u)\right]$	[0.015, 0.0329]	[0.0166, 0.0289]

Step-8: Calculate closeness coefficients using q-rung NVSS-PIS and q-rung NVSS-NIS for every alternatives. The C^* values are $C_1^* = 0.7173$, $C_2^* = 0.6281$, $C_3^* = 0.628$, $C_4^* = 0.6373$, $C_5^* = 0.6983$, $C_6^* = 0.5196$, $C_7^* = 0.5834$, $C_8^* = 0.6825$, $C_9^* = 0.7148$, $C_{10}^* = 0.5912$.

Step-9: The order of the diagnostic health imaging C_i^* is $\chi_1 \succeq \chi_9 \succeq \chi_5 \succeq \chi_8 \succeq \chi_4 \succeq \chi_2 \succeq \chi_3 \succeq \chi_{10} \succeq \chi_7 \succeq \chi_6$.

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

5 Comparison between the suggested and the existing approach

Here, we compared some existing models with the suggested models. Thus, its benefits and usefulness are demonstrated. A q-rung NVSS approach is used with an ED. Here are the various distances:

$q=1$	q-rung NVSS
$TOPSIS(proposed)$	$\chi_1 \succeq \chi_9 \succeq \chi_5 \succeq \chi_8 \succeq \chi_4 \succeq \chi_2 \succeq \chi_3 \succeq \chi_{10} \succeq \chi_7 \succeq \chi_6$
$TOPSIS$-[17]	$\chi_1 \succeq \chi_9 \succeq \chi_5 \succeq \chi_8 \succeq \chi_4 \succeq \chi_2 \succeq \chi_3 \succeq \chi_{10} \succeq \chi_7 \succeq \chi_6$
$TOPSIS$-[16]	$\chi_1 \succeq \chi_9 \succeq \chi_5 \succeq \chi_8 \succeq \chi_4 \succeq \chi_2 \succeq \chi_3 \succeq \chi_{10} \succeq \chi_7 \succeq \chi_6$

5.1 Sensitivity Analysis

Comparing the MCGDM approach with alternatives in terms of reliability of the circumstances. Closeness values and rankings are listed below. A q-rung NVSS technique is used to acquire the various q values. Using the q-rung NVSS method, if $q = 1$, then ranking of alternative is $\chi_1 \succeq \chi_9 \succeq \chi_5 \succeq \chi_8 \succeq \chi_4 \succeq \chi_2 \succeq \chi_3 \succeq \chi_{10} \succeq \chi_7 \succeq \chi_6$. If $2 \leq q \leq 4$, then rearrange ranking is $\chi_1 \succeq \chi_5 \succeq \chi_9 \succeq \chi_8 \succeq \chi_4 \succeq \chi_3 \succeq \chi_2 \succeq \chi_7 \succeq \chi_{10} \succeq \chi_6$. If $q = 5$, then ranking of new order is $\chi_5 \succeq \chi_1 \succeq \chi_9 \succeq \chi_8 \succeq \chi_4 \succeq \chi_3 \succeq \chi_2 \succeq \chi_7 \succeq \chi_{10} \succeq \chi_6$. Thus, the "$\chi_1$" is changed to "$\chi_5$" as the best alternative. There should be a change to TOPSIS values for q. Different values of q are as follows:

Different approach	q-rung NVSS
$q=2$	$\chi_1 \succeq \chi_9 \succeq \chi_5 \succeq \chi_8 \succeq \chi_4 \succeq \chi_3 \succeq \chi_2 \succeq \chi_{10} \succeq \chi_7 \succeq \chi_6$
$q=3$	$\chi_1 \succeq \chi_9 \succeq \chi_8 \succeq \chi_5 \succeq \chi_4 \succeq \chi_3 \succeq \chi_2 \succeq \chi_7 \succeq \chi_{10} \succeq \chi_6$
$q=4$	$\chi_1 \succeq \chi_5 \succeq \chi_9 \succeq \chi_8 \succeq \chi_4 \succeq \chi_3 \succeq \chi_2 \succeq \chi_7 \succeq \chi_{10} \succeq \chi_6$
$q=5$	$\chi_5 \succeq \chi_1 \succeq \chi_9 \succeq \chi_8 \succeq \chi_4 \succeq \chi_3 \succeq \chi_2 \succeq \chi_7 \succeq \chi_{10} \succeq \chi_6$

Figure 1 shows the graphical representation consisting of MCGDM based on TOPSIS.

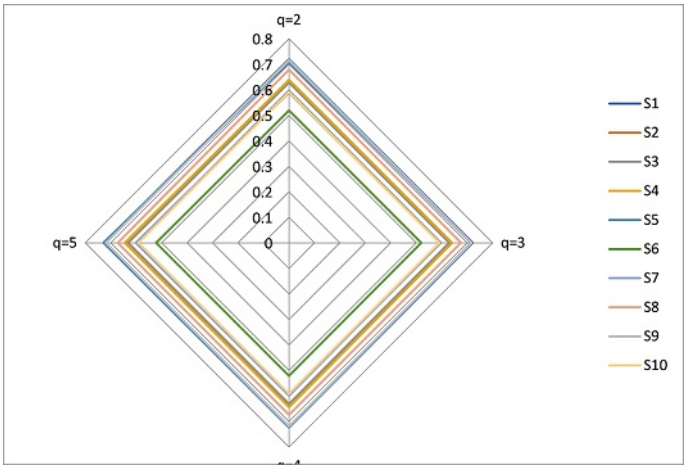

Figure 1: Graphical representation using MCGDM based on TOPSIS.

5.2 Advantages

Several advantages can be seen in the technique recommended in the following paragraphs based on the earlier study described above. It proposes the idea of q-rung NVSS by combining FS and VNSS. q-rung NVSS analyzes human behaviour and natural phenomena that take place in real life and clarifies ambiguous information when TV, IV and FV sum to more than two but less than one in the case of the square sum of their TV, IV, and FV. A decision maker can choose the outcome based on his or her personal preferences and q. A q-rung NVSS provides flexibility in achieving varied rankings of alternatives.

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

6 Conclusion

This article introduced ED for q-rung NVSSs, which have the advantage of being mathematically simple. There are numerous numerical examples that illustrate the superiority of ED. The usefulness of ED is illustrated through an application case. The main objective of this research is to develop a q-rung NVSS based on the challenges of DM. We reached a number of conclusions regarding q-rung NVSS of numerous AO. Using the q-rung NVSS based on the MCGDM method, people can make decisions based on ambiguous and conflicting information. We have applied the operators associated with q-rung NVSS to the MCGDM problem based on q. The ranking of alternatives can be determined using a q rung NVSS. Generally, a generalized q value affects ranking alternatives most significantly. It is possible to arrive at the most reasonable ranking by setting q values in accordance with the actual scenario. The decision-maker must know q to be able to determine the result. However, this paper aims to serve as a foundation for future academics who are interested in this field of study despite its infancy. We have also included a variety of statistical charts to illustrate how the alternatives rank.

Acknowledgments: This research project (Fuzzy Algebras and Applications of Fuzzy Soft Matrices in Decision-Making Problems) was supported by the Thailand Science Research and Innovation Fund and the University of Phayao (Grant No. FF67-UoE-Aiyared-Iampan).

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] T. M. Al-Shami, H. Z. Ibrahim, A. A. Azzam, and A. I. EL-Maghrabi, SR-fuzzy sets and their weighted aggregated operators in application to decision-making, Journal of Function Spaces, 2022, (2022), Article ID 3653225, 14 pages.

[2] K. Atanassov, Intuitionistic fuzzy sets, Fuzzy Sets and Systems, 20(1), (1986), 87-96.

[3] R. Biswas, Vague groups, International Journal of Computational Cognition, 4(2), (2006), 20-23.

[4] S. Broumi, S. Krishna Prabha, and V. Uluçay, Interval-valued Fermatean neutrosophic shortest path problem via score function, Neutrosophic Systems with Applications, 11, (2023), 1–10.

[5] S. Broumi, S. Mohanaselvi, T. Witczak, M. Talea, A. Bakali, and F. Smarandache, Complex Fermatean neutrosophic graph and application to decision making, Decision Making: Applications in Management and Engineering, 6(1), (2023), 474-501.

[6] S. Broumi, P. K. Raut, and S. P. Behera, Solving shortest path problems using an ant colony algorithm with triangular neutrosophic arc weights, International Journal of Neutrosophic Science, 20(4), (2023), 128-28.

[7] S. Broumi, R. Sundareswaran, M. Shanmugapriya, A. Bakali, and M. Talea, Theory and applications of Fermatean neutrosophic graphs, Neutrosophic Sets and Systems, 50, (2022), 248-286.

[8] S. Broumi, R. Sundareswaran, M. Shanmugapriya, P. K. Singh, M. Voskoglou, and M. Talea, Faculty performance evaluation through multi-criteria decision analysis using interval-valued Fermatean neutrosophic sets, Mathematics, 11(18), (2023), 3817.

[9] B. C. Cuong and V. Kreinovich, Picture fuzzy sets a new concept for computational intelligence problems, Third world congress on information and communication technologies, WICT 2013, IEEE, 1-6.

[10] P. A. Ejegwa, Distance and similarity measures for Pythagorean fuzzy sets, Granular Computing, 5, (2020), 225–238.

[11] R. Jansi, K. Mohana, and F. Smarandache, Correlation measure for Pythagorean neutrosophic sets with T and F as dependent neutrosophic components, Neutrosophic Sets and Systems, 30, (2019), 202-212.

[12] B. P. Joshi, A. Singh, P. K. Bhatt, and K. S. Vaisla, Interval valued q-rung orthopair fuzzy sets and their properties, Journal of Intelligent and Fuzzy Systems, 35, (2018), 5225-5230.

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023

[13] P. K. Maji, R. Biswas, and A. R. Roy, Fuzzy soft sets, Journal of Fuzzy Mathematics, 9(3), (2001), 589-602.

[14] P. K. Maji, R. Biswas, and A. R. Roy, On intuitionistic fuzzy soft sets, Journal of Fuzzy Mathematics, 9(3), (2001), 677-692.

[15] D. Molodtsov, Soft set theory-First results, Computers and Mathematics with Applications, 37(4-5), (1999), 19-31.

[16] M. Palanikumar and K. Arulmozhi, MCGDM based on TOPSIS and VIKOR using Pythagorean neutrosophic soft with aggregation operators, Neutrosophic Sets and Systems, 51, (2022), 538-555.

[17] M. Palanikumar, K. Arulmozhi, and A. Iampan, Multi criteria group decision making based on VIKOR and TOPSIS methods for Fermatean fuzzy soft with aggregation operators, ICIC Express Letters, 16(10), (2022), 1129-1138.

[18] M. Palanikumar, K. Arulmozhi, A. Iampan, and S. Broumi, Medical diagnosis decision making using type-II generalized Pythagorean neutrosophic interval valued soft sets, International Journal of Neutrosophic Science, 20(1), (2023), 85-105.

[19] M. Palanikumar, K. Arulmozhi, A. Iampan, and L. J. Manavalan, Novel possibility Pythagorean cubic fuzzy soft sets and their applications, International Journal of Innovative Computing, Information and Control, 19(2), (2023), 325-337.

[20] M. Palanikumar and S. Broumi, Square root Diophantine neutrosophic normal interval-valued sets and their aggregated operators in application to multiple attribute decision making, International Journal of Neutrosophic Science, 19(3), (2022), 63-84.

[21] M. Palanikumar, A. Iampan, S. Broumi, and G. Balaji, Generalization of neutrosophic interval-valued soft sets with different aggregating operators using multi-criteria group decision-making, International Journal of Neutrosophic Science, 22(1), (2023), 114-123.

[22] M. Palanikumar, A. Iampan, S. Broumi, L. J. Manavalan, and K. Sundareswari, Multi-criteria group decision making method in Pythagorean interval-valued neutrosophic fuzzy soft soft using VIKOR approach, International Journal of Neutrosophic Science, 22(1), (2023), 104-113.

[23] M. Palanikumar, N. Kausar, H. Garg, A. Iampan, S. Kadry, and M. Sharaf, Medical robotic engineering selection based on square root neutrosophic normal interval-valued sets and their aggregated operators, AIMS Mathematics, 8(8), (2023), 17402-17432.

[24] X. Peng and Y. Yang, Fundamental properties of interval valued Pythagorean fuzzy aggregation operators, International Journal of Intelligent Systems, 31(5), (2016), 444-487.

[25] F. Smarandache, A unifying field in logics, Neutrosophy neutrosophic probability, set and logic, American Research Press, Rehoboth, 1999.

[26] R. R. Yager, Generalized orthopair fuzzy sets, IEEE Transactions on Fuzzy Systems, 25(5), (2016), 1222-1230.

[27] R. R. Yager, Pythagorean membership grades in multi criteria decision making, IEEE Trans. Fuzzy Systems, 22, (2014), 958-965.

[28] L. A. Zadeh, Fuzzy sets, Information and control, 8(3), (1965), 338-353.

https://doi.org/10.54216/IJNS.220406

Received: May 28, 2023 Revised: July 09, 2023 Accepted: November 01, 2023