Computers, Materials & Continua
DOI: 10.32604/cmc.2023.033175
Article

Type-2 Neutrosophic Set and Their Applications in Medical Databases Deadlock Resolution

Marwan H. Hassan[1], Saad M. Darwish[2,*] and Saleh M. Elkaffas[3]

[1]Department of Advocacy and Public Speaking, Imam Al-Adham University College, Anbar, 55431, Iraq
[2]Department of Information Technology, Institute of Graduate Studies and Research, Alexandria University, Alexandria, 21526, Egypt
[3]College of Computing and Information Technology, Arab Academy for Science, Technology and Maritime Transport, Alexandria, 1029, Egypt
*Corresponding Author: Saad M. Darwish. Email: saad.darwish@alexu.edu.eg
Received: 09 June 2022; Accepted: 09 September 2022

Abstract: Electronic patient data gives many advantages, but also new difficulties. Deadlocks may delay procedures like acquiring patient information. Distributed deadlock resolution solutions introduce uncertainty due to inaccurate transaction properties. Soft computing-based solutions have been developed to solve this challenge. In a single framework, ambiguous, vague, incomplete, and inconsistent transaction attribute information has received minimal attention. The work presented in this paper employed type-2 neutrosophic logic, an extension of type-1 neutrosophic logic, to handle uncertainty in real-time deadlock-resolving systems. The proposed method is structured to reflect multiple types of knowledge and relations among transactions' features that include validation factor degree, slackness degree, degree of deadline-missed transaction based on the degree of membership of truthiness, degree of membership of indeterminacy, and degree of membership of falsity. Here, the footprint of uncertainty (FOU) for truth, indeterminacy, and falsity represents the level of uncertainty that exists in the value of a grade of membership. We employed a distributed real-time transaction processing simulator (DRTTPS) to conduct the simulations and conducted experiments using the benchmark Pima Indians diabetes dataset (PIDD). As the results showed, there is an increase in detection rate and a large drop in rollback rate when this new strategy is used. The performance of Type-2 neutrosophic-based resolution is better than the Type-1 neutrosophic-based approach on the execution ratio scale. The improvement rate has reached 10% to 20%, depending on the number of arrived transactions.

Keywords: Deadlock recovery; type-2 neutrosophic set; healthcare databases; distributed deadlock detection

1 Introduction

Online healthcare solutions give constant access to medical treatments. These systems rely on a network to share data between patients, doctors, and nurses. This leads to huge volumes of medical data, including procedures, medicines, and allergies. Properly constructed clinical information systems enable doctors to access patient information as needed. Computer freezes may create information delays for electronic system users. A deadlock is a common cause of a computer's seeming to freeze. During the health information management service (HIMS) pilot phase, and as the number of users increased, the necessity to handle deadlock and concurrency issues became prominent [1–3]. Several parameters are examined while creating a distributed database, including processor location, data availability and dependability, workload dispersion, and storage costs *vs.* availability. Using all of these parameters together might result in sophisticated optimization models. Thus, it's common to prioritize the most important objectives, such as maximizing processing locality, and regard the rest as constraints [4]. Recently, real-time database systems (RTDBS) have gained popularity. RTDBS is a relational database management system with deadlines. RTDBS's effectiveness and accuracy depend on meeting these deadlines [5,6]. In distributed real-time database systems (DRTDBS), blocked transactions miss deadlines.

One of the most common problems in operating systems and databases is resolving deadlocks. Additionally, the system's resource consumption and the events connected to deadlock processes are affected [7]. Distributed databases allow several users to simultaneously access the same database at the same time, which is called a "concurrent transaction". Time stamp ordering-based concurrency control and optimistic concurrency control are a few of the concurrency control strategies used to achieve this synchronization [8]. There are four types of deadlock in a transaction: mutual exclusion, hold and wait, no preemption, and circular waiting. An oriented graph known as a system resource allocation graph may precisely characterize a deadlock [9]. The three ways to deal with deadlocks are to prevent, avoid, or resolve deadlocks that have already occurred in transactional systems. In order to avoid deadlocks, the system as a whole must be designed to make them impossible [10]. Avoiding deadlocks in real time means allocating resources and initiating transactions in a way that prevents them at runtime [11]. When a deadlock is found and resolved by utilizing a sequence of protocols such as monitoring the wait-for-graph (WFG) of the current transactions or sending a "probe" to locate cycles, resolution implies that deadlocks are permitted [12].

Hard (rigid deadline) and soft (flexible deadline) represent a new breed of real-time, distributed databases that combine distributed technologies to fulfil both performance and accuracy requirements. The main system must be developed to deal with deadlocks in an efficient way without adding unnecessary complexity. The fuzziness of transaction object properties might lead to false positives and failures in the deadlock detection process since these techniques don't take this into consideration. It's also difficult to tell which deadlock is most critical, since these approaches don't reveal the importance of all of them [13,14]. A wide variety of applications rely on fuzzy logic systems to accurately describe uncertainty. What this means in practice is that the approximate and imprecise aspects of reality may be accurately represented. Confidence or uncertainty in a set's membership is quantified using fuzzy logic. There have been a lot of attempts made to use these sets to reduce the imprecision in such cases [14].

Real-world situations need to deal with uncertainty, yet fuzzy sets are unable to deal with indeterminacy in data. A neutrosophic set was therefore introduced [15,16] as a result. Neutrosophic sets were established to cope with greater ambiguity in the situation by allowing experts or individuals who answered a questionnaire to contribute more knowledge and data that was more indeterminate. The membership functions of the three functions, truth (T), indeterminacy (I), and falsity (F), are

not uncertain in a neutrosophic set. Consequently, it is unable to cope with information in the form of words and sentences in artificial languages known as linguistic variables, which minimizes the total computing complexity of any real-world issue. A type-2 neutrosophic set (T2NS) may reduce uncertainty and indeterminacy in real-world issues better than other sets because it's FOU indicates its degree of uncertainty (see Fig. 1). It is feasible to attach alternative language variables to the truth, indeterminacy, and falsity membership functions since they are independent. Because of this, T2NS has an advantage [17–20]. By including a T2NS into the model, fewer risky judgments may be made [21,22].

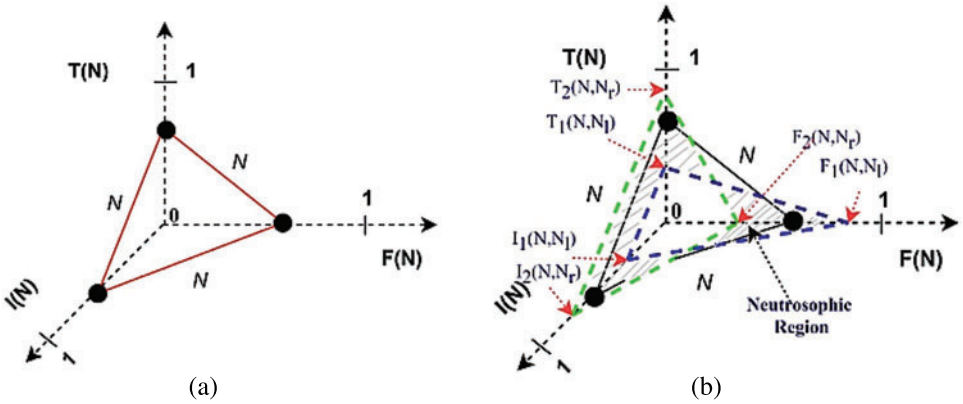

Figure 1: (a) Neutrosophic truth, indeterminacy, and falsity membership functions (b) Type-2 neutrosophic set membership functions [19]

Using type2-neutrosophic data, our goal is to resolve deadlocks based on attributes that are ambiguous in nature. Type-2 Neutrosophic logic introduced an FOU-based indeterminacy concept, which is used in the proposed model. This mechanism uses transaction features that include the degree of deadline-missed transaction, validation factor degree, and slackness degree to resolve deadlocks between transactions whenever conflict arises, thereby decreasing transaction re-executions or waiting and the current load on the database server. Type-2 neutrosophic logic has not been used previously in resolving deadlocks in distributed database systems.

The remainder of this paper is organized as follows: in Section 2, we analyze the most important deadlock control solutions discussed in the literature. Following that, Section 3 presents the proposed model by demonstrating and describing in detail the model tasks. In Section 4, we evaluate the performance of the recommended technique. Finally, Section 5 summarizes our results and outlines our future intentions.

2 Related Work

Distributed deadlocks may be detected using a variety of methods [8,10,23]. The avoid deadlock strategy is used when there is a substantial likelihood of a deadlock. The detection and resolved-deadlock approach, on the other hand, is used when the likelihood of a deadlock is minimal. This section highlights scholars' contributions and deadlock-resolution solutions. The work presented in [8] describes a deadlock detection and resolution mechanism based on transaction priority. The highest-priority initial cycle transaction is recorded in a priority table. Priority-based tables overcome deadlocks. The lowest-priority transaction is cancelled to free up resources for waiting transactions. This approach fails when priorities change.

In [24], the transaction wait for graph (TWFG) method was utilized to design a unique way to resolve priority changes by constructing two structures: a local transaction structure for identifying local deadlock and a global transaction structure for detecting distributed deadlock. TWFG reduces the requirement for local distributed deadlock detection. One probe message is sent on each WFG edge, hence deadlocks are identified slowly. These methods can identify and resolve deadlock in distributed databases, but they have constraints, including priority, standard criteria, and starvation. In [25], the authors developed a novel way to handle the priority change issue by combining the TWFG algorithm for deadlock detection with Grover's technique for determining the victim transaction using a time stamp.

The transaction degree is a deadlock-resolution mechanism presented in [26]. This solves deadlock by aborting the transaction with the most out-of-degree and in-degree. The authors in [23] presented a simple method for detecting deadlock. This simple method uses an update message with two functions: one modifies Wait-for variables and the other checks for deadlock. This method doesn't figure out which transactions should be stopped early to avoid deadlocks and the costs that come with them. In [27], a distributed deadlock blocking solution was suggested that uses previous knowledge of necessary resources by extending the two-phase commit procedure. Thread-specific partitions make lock dependencies accessible. The lock dependence set search reduces related permutations. This method eliminates lock dependencies before deadlock localization.

In [28], a scalable and extendable model of two phase locking (2PL)-concurrency control techniques based on hierarchical colored petri nets was developed. State space analysis determines whether all transaction schedules are deadlock-free. The model can easily replicate and analyze concurrency control techniques like strict 2PL. How to run transactions seeking the same resources in a pipeline to prevent deadlocks and minimize waiting time has been addressed [29]. The system described in [14] used similarity between conflicting activities to boost real-time performance and transaction criticality to prioritize data conflict resolution. The system used fuzzy logic, a well-known artificial intelligence (AI) concept, to integrate transaction attributes to resolve conflicts. In [9], the authors suggested a method for overcoming deadlocks that combines fuzzy and Aristotelian logic with logical concepts. Mamdani's controller-based technique breaks the deadlock. It decreases the chance of deadlocks, then finds and resolves them.

This article extends the type-1 neutrosophic-based deadlock handling approach with type-2 neutrosophic logic. Neutrosophic systems use human knowledge like fuzzy ones. A fuzzy set uses the membership grade to handle uncertainty, whereas a type-2 neutrosophic set uses FOU-based independent truth, indeterminacy, and falsity membership grades. Type-2 neutrosophic logic captures uncertainty well and delivers realistic membership grades. This new theory provides a granular representation of transaction features and helps to model uncertainties with six different memberships very effectively.

3 The Proposed Method

Using type-2 neutrosophic logic, we propose a new model for resolving deadlocks that takes into account the uncertainty of transaction characteristics. The Neutrosophic number is concerned largely with ambiguous, incomplete, and inconsistent data. This theory is an extension of type-1 neutrosophic concepts since each element of a neutrosophic set has two membership functions for truth, indeterminacy, and falsity. In this research article, our primary purpose is to explore the type-2 neutrosophic logic concept in resolving deadlocks in complex uncertain distributed database systems through: (1) expressing input parameters (antecedents) in type-2 trapezoidal neutrosophic numbers,

(2) constructing neutrosophic IF-THEN rules to estimate the deadlock resolving using AND operator to model qualitative features of human understanding, (3) deneutrosophication of the consequents and comparing them with the detection rate and rollback rate to arrive at the throughput of the system.

The suggested technique has the following advantages: (1) Access priority is maintained to assure serializability without aborting transactions. (2) Transaction execution time costs less than re-execution. A transaction may wait longer to obtain data rather than access it and abort it. (3) The transaction with the greatest work is prioritized since it will be finished faster with greater rights. (4) By giving a little waiting time, system throughput increases and overhead is lowered. Table 1 shows the method's parameters. Herein, transaction managers have the following responsibilities: (1) Transaction delimitation: start, commit, and rollback. (2) Transaction contexts provide all the information a transaction manager needs to monitor a transaction. Transaction managers create and associate transaction contexts with threads. (3) Transaction managers may commonly coordinate a transaction across several resources. (4) Failure recovery: transaction managers ensure resources aren't left in an inconsistent state after a system or application failure.

Table 1: List of parameters used in the proposed approach

Parameter	Definition
$D\,(T)$	Transaction's data object
T	Transaction
$s\,(T)$	Start time of transaction T
$d\,(T)$	deadline of transaction T
$VF_t\,(T)$	Validation factor of transaction T at time t
$DLMT$	Total number of deadline-missing transactions
S	Slackness degree
N	Total number of transactions processed
t_a	The arrival time of a transaction
l	Amount of unfinished work
c	Amount of computation already invested
TS	Deadlock's transaction status
MP	Miss percentage
t	Current time

3.1 Step 1: Transaction Reading

The resource manager accepts transaction requests and specifies the requisite resources and lock mode for each transaction (see Fig. 2) [30]. It also sorts transactions by timestamp. Each read-phase transaction receives a timestamp interval. This interval stores a transaction's temporary serialization order [31]. Each transaction is identified by a unique transaction identifier. However, changing the identifier may cause the order of transactions to change, e.g., an older transaction may become the youngest. To circumvent this, the new system associates each transaction with a timestamp (in addition to the identification) showing the time the transaction entered the system [32]. This timestamp is not affected after an abort, and so may be utilized for transaction ordering. For simplicity reasons, we will utilize identifiers to arrange transactions and will assume that they have such a timestamp [33]. During

the validation phase of transaction T_i, the technique verifies that T_i does not interact with any other committed transactions or currently validating transactions. Both conflict detection and resolution occur during the validation phase of a transaction's execution. Reducing the number of transaction restarts is a critical strategy to enhance the speed of deadlock resolution solutions [34,35].

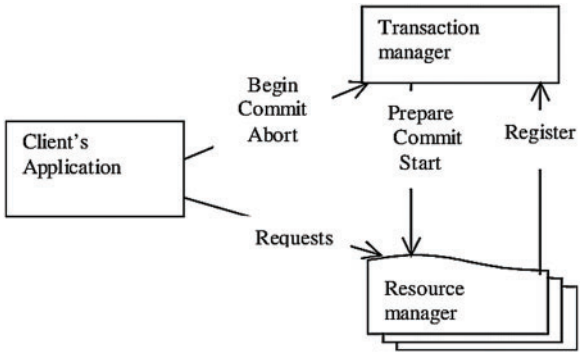

Figure 2: Database transaction processing system

3.2 Step 2: Distributed Deadlock Detection

The concept of a *WFG* was employed to discover deadlocks [36]. A system is said to be in deadlock if and only if it encounters a cycle in the *WFG*. Then, each transaction in the *WFG* that is a part of the cycle is considered as though it is in a deadlock condition (see Fig. 3). If a system encounters a deadlock, the next stage is to recover via neutrosophic mechanisms [37]. To handle distributed database deadlock detection, as in our case, the proposed model employs an optimized path-pushing strategy, which involves sending a portion of a possible cycle, referred to as a "path" to another site only when the first transaction in the path has a higher priority than the last one [38]. This results in a halving of the quantity of messages [39]. The approach identifies phantom deadlocks because the portions of the *WFG* sent between sites are asynchronously captured snapshots, i.e., they might be inconsistent. For further information, see [40,41].

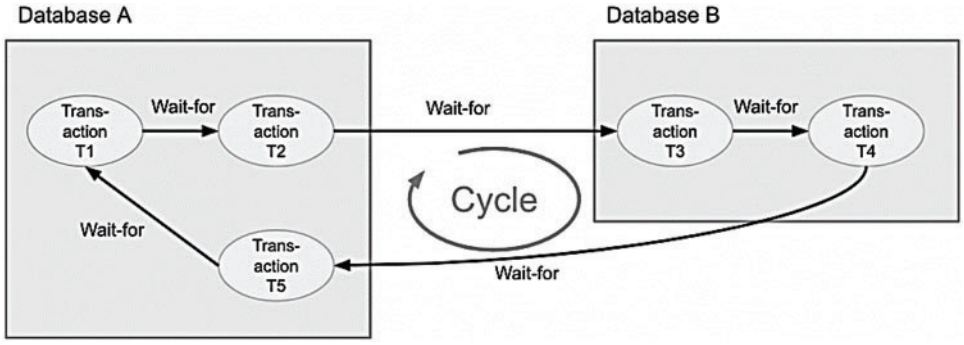

Figure 3: Global wait-for-graph with a cycle

3.3 Step 3: Extracting Transaction Attributes

In general, existing techniques to resolve deadlock in RTDBS rely on transaction scheduling (serialization) that is motivated by priority rather than fairness concerns. See [42] for the drawbacks of

the serialization technique. As discussed before, a deadlock avoidance method needs basic knowledge about the transaction structure and the required resources, yet this information is often unavailable or imprecise [43]. Thus, the proposed technique in this paper is capable of resolving deadlock issues via the use of a simple structure that exploits transactional properties and neutrosophic logic to handle the vague nature of these properties. In our situation, deadlock resolution in the RTDBS is determined by three transaction characteristics [44]: the degree of deadline-missing transactions, the degree of validation factor, and the degree of slackness [45].

- *Degree of deadline-missed transaction*: the primary objective of RTDBS is to adhere to the time limits imposed by the activities. As a result, the major performance metric is the "miss percentage," or the proportion of transactions that miss their deadlines. The following equation is used to determine the proportion of data that is missing [38]:

$$Miss\ Percentage\ (MP) = 100 * DLMT/N \tag{1}$$

The smaller the proportion of transactions that miss their deadline, the greater the chance of waiting for the transaction.

- *Validation factor* (VF) *degree*: The suggested technique is designed in such a way that a checking algorithm is used to ensure that validated data is used in conjunction with the transaction scheduling process. The checking process guarantees that all temporal data in a transaction's read set stays valid throughout its execution time, hence ensuring the transaction's temporal consistency. Following that, the key factor concurrency control method updates validation rules during the validation phase, which schedules near-completed priority transactions by asserting the validation factor. The validation algorithm determines the transaction's validation factor, which is a variable computed from the current time, the transaction's start time, and the transaction's deadline time. The degree of validation is computed using the following equation [38]:

$$VF_t\left(T_1\right) = \left(t - s\left(T_1\right)\right) / \left(d\left(T_1\right) - s\left(T_1\right)\right) \tag{2}$$

The lower the validation factor degree, the longer the deadline is, the probability of waiting for the transaction increases.

- *Slackness degree*: Slackness quantifies the amount of time that a transaction's execution may be delayed while still meeting the transaction's deadline. If we use the term t_a to signify the arrival time of a transaction, slackness (S) may be stated as [44]:

$$S = d\left(T\right) - t_a - c - l \tag{3}$$

The more slackness a transaction has, the higher its priority should be. In this situation, the earlier the transaction's deadline, the greater its priority. A transaction with a smaller quantity of unfinished work may be prioritized over one with a large amount of unfinished work. When a transaction enters the commit phase, its priority may be increased to a higher value. This allows a committed transaction with a minimum processing time to be completed quickly. Thus, resources held by the committed transaction may be freed sooner, preventing subsequent transactions from being blocked. A transaction that has already completed a significant amount of computation may be assigned a higher priority. This procedure shortens turnaround time and aids in maintaining external consistency of data. Prior knowledge of MP, VF, and S is necessary to determine the deadlock. Human beings are more at ease making judgments on linguistic variables. Linguistic variables enable us to translate imprecise ideas stated in natural languages into exact mathematical expressions. Due to the high degree

of fuzziness and uncertainty inherent in linguistic variables, neutrosophic sets are well suited to solving issues involving such data [21,22]. Fig. 4 illustrates the process of feature extraction.

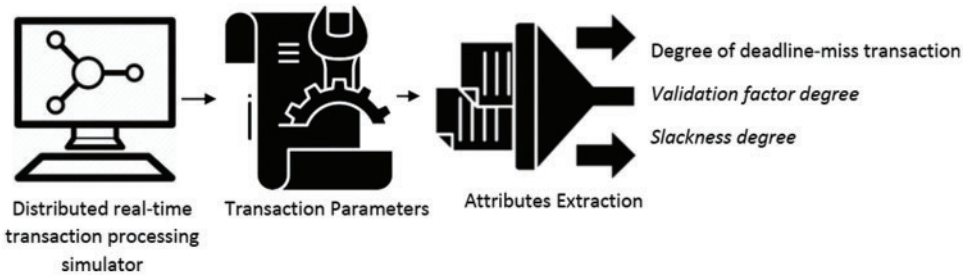

Figure 4: Transaction features extraction process

3.4 Step 4: Type-2 Neutrosophication

Our suggested technique employs type-2 trapezoidal neutrosophic numbers with linear membership functions. These membership curves are not mutually exclusive, and neighboring curves may overlap in certain areas. Experts determine their input values based on a comparison of nearby membership curves' properties. We get membership degrees for the input values depending on which part of the membership curve these values fall inside [46]. At this stage, the inputs (attributes) *MP*, *VF*, and *S* are extracted in crisp numerical form within the universe of discourse by running the RTDBS simulator for medical databases. Membership functions are used to determine the degree of membership of each input to the proper neutrosophic set. The degree of membership is between 0 and 1. In our scenario, they all have three linguistic levels, such as "Low," "Medium," and "High," with corresponding linear membership functions. *DS* expressions are also given in linguistic variables that include 'wait' and 'execute'. See [46] for the definition of the linear trapezoidal neutrosophic function. The range and neutrosophic set definitions $a_1'' \le a_1 \le a_1' \le a_2 \le a_3 \le a_4' \le a_4 \le a_4''$ (see Fig. 5) for each attribute are given based on the maximum and minimum values of this variable by running the RTDBS simulator. Fig. 6 illustrates the graphical representation of the type-2 neutrosophic membership function.

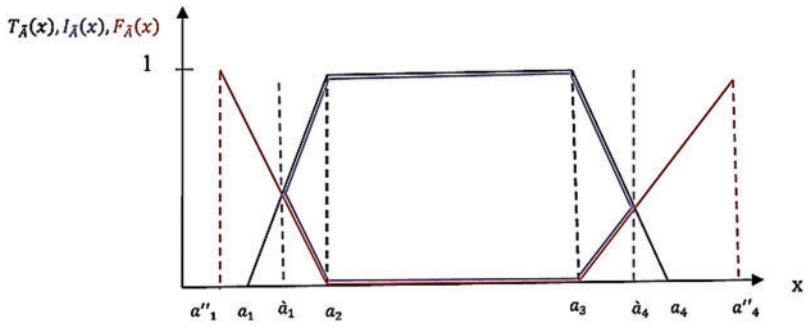

Figure 5: Linear trapezoidal neutrosophic number [46]

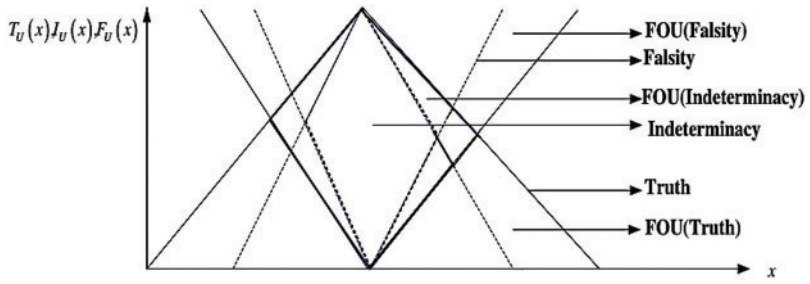

Figure 6: Graphical representation of type-2 neutrosophic membership function [20]

A single-valued neutrosophic set (SVNS) $\breve{S}$ on universal set U is characterized by truth membership function TMF $\left(\varnothing_{\breve{S}}\right)$, indeterminacy membership function IMF $\left(\varnothing_{\breve{S}}\right)$ and falsity membership function FMF $\left(\varnothing_{\breve{S}}\right)$ respectively, in the following way [19]:

$$\breve{S} = \left\{\langle \xi, \left(\varnothing_{\breve{S}}(\xi), \psi_{\breve{S}}(\xi), \varphi_{\breve{S}}(\xi)\right)\rangle : \xi \in U, \varnothing_{\breve{S}}(\xi), \psi_{\breve{S}}(\xi), \varphi_{\breve{S}}(\xi) \in [0,1]\right\} \tag{4}$$

Such that $0 \leq \varnothing_{\breve{S}}(\xi), \psi_{\breve{S}}(\xi), \varphi_{\breve{S}}(\xi) \leq 3$. Let $\breve{S}(\xi) = \left[\breve{S}^{U}(\xi), \breve{S}^{L}(\xi)\right]$ be an interval type-2 neutrosophic set (IT2NS) on universal set U where $\xi \in$ U and $\breve{S}^{U} : \mathrm{U} \to [0,1]$ and $\breve{S}^{L} : \mathrm{U} \to [0,1]$ are two type-1 neutrosophic sets (T1NSs) known as upper and lower neutrosophic sets respectively having the condition $0 \leq \breve{S}^{L}(\xi) \leq \breve{S}^{U}(\xi) \leq 1$ defined as follows [19]:

$$\begin{aligned}\breve{S} = \Big\{\langle \xi, \big(&\left[\varnothing_{\breve{S}}{}^{U}(\xi), \varnothing_{\breve{S}}{}^{L}(\xi)\right], \left[\psi_{\breve{S}}{}^{U}(\xi), \psi_{\breve{S}}{}^{L}(\xi)\right], \left[\varphi_{\breve{S}}{}^{U}(\xi), \varphi_{\breve{S}}{}^{L}(\xi)\right]\big)\rangle : \xi \in U\Big\},\\ &\left[\varnothing_{\breve{S}}{}^{U}(\xi), \varnothing_{\breve{S}}{}^{L}(\xi)\right], \left[\psi_{\breve{S}}{}^{U}(\xi), \psi_{\breve{S}}{}^{L}(\xi)\right], \left[\varphi_{\breve{S}}{}^{U}(\xi), \varphi_{\breve{S}}{}^{L}(\xi)\right] \in [0,1]\end{aligned} \tag{5}$$

$$\varnothing_{\breve{S}}(\xi) = \begin{cases} \dfrac{\left(\xi - \breve{S}_1\right)\varnothing_{\breve{S}}}{\breve{S}_2 - \breve{S}_1} & \breve{S}_1 \leq \xi \leq \breve{S}_2 \\ \varnothing_{\breve{S}} & \breve{S}_2 \leq \xi \leq \breve{S}_3 \\ \dfrac{\left(\breve{S}_4 - \xi\right)\varnothing_{\breve{S}}}{\breve{S}_4 - \breve{S}_3} & \breve{S}_3 \leq \xi \leq \breve{S}_4 \\ 0 & otherwise \end{cases} \tag{6}$$

$$\psi_{\breve{S}}(\xi) = \begin{cases} \dfrac{\breve{S}_2 - \xi + \left(\xi - \breve{S}_1\right)\psi_{\breve{S}}}{\breve{S}_2 - \breve{S}_1} & \breve{S}_1 \leq \xi \leq \breve{S}_2 \\ \psi_{\breve{S}} & \breve{S}_2 \leq \xi \leq \breve{S}_3 \\ \dfrac{\xi - \breve{S}_3 + \left(\breve{S}_4 - \xi\right)\psi_{\breve{S}}}{\breve{S}_4 - \breve{S}_3} & \breve{S}_3 \leq \xi \leq \breve{S}_4 \\ 1 & otherwise \end{cases} \tag{7}$$

$$\varphi_{\breve{S}}(\xi)=\begin{cases}\dfrac{\breve{S}_2-\xi+\left(\xi-\breve{S}_1\right)\varphi_{\breve{S}}}{\breve{S}_2-\breve{S}_1} & \breve{S}_1\le\xi\le\breve{S}_2\\ \varphi_{\breve{S}} & \breve{S}_2\le\xi\le\breve{S}_3\\ \dfrac{\xi-\breve{S}_3+\left(\breve{S}_4-\xi\right)\varphi_{\breve{S}}}{\breve{S}_4-\breve{S}_3} & \breve{S}_3\le\xi\le\breve{S}_4\\ 1 & otherwise\end{cases} \tag{8}$$

where $\varnothing_{\breve{S}}=\left[\varnothing_{\breve{S}}{}^{U},\varnothing_{\breve{S}}{}^{L}\right],\psi_{\breve{S}}=\left[\psi_{\breve{S}}{}^{U},\psi_{\breve{S}}{}^{L}\right]$ and $\varphi_{\breve{S}}=\left[\varphi_{\breve{S}}{}^{U},\varphi_{\breve{S}}{}^{L}\right]$ are interval type-2 neutrosophic numbers (IT2NNs). The number $\breve{S}$ can be represented as (see Fig. 7):

$$\breve{S}=\left[\breve{S}^{U},\breve{S}^{L}\right]=\left[\left(\breve{S}_1^{U},\ \breve{S}_2^{U},\ \breve{S}_3^{U},\breve{S}_4^{U};\varnothing_{\breve{S}}^{U},\psi_{\breve{S}}^{U},\varphi_{\breve{S}}^{U}\right),\left(\breve{S}_1^{L},\ \breve{S}_2^{L},\ \breve{S}_3^{L},\breve{S}_4^{L};\varnothing_{\breve{S}}^{L},\psi_{\breve{S}}^{L},\varphi_{\breve{S}}^{L}\right)\right] \tag{9}$$

and is called interval type-2 trapezoidal neutrosophic logic number (IT2TrNN) where

$$0\le\breve{S}_1^{U}\le\breve{S}_2^{U}\le\breve{S}_3^{U}\le\breve{S}_4^{U}\le1,\ \ 0\le\breve{S}_1^{L}\le\breve{S}_2^{L}\le\breve{S}_3^{L}\le\breve{S}_4^{L}\le1,$$
$$0\le\varnothing_{\breve{S}}^{L}\le\varnothing_{\breve{S}}^{U}\le1,\ 0\le\psi_{\breve{S}}^{L}\le\psi_{\breve{S}}^{U}\le1,\ 0\le\varphi_{\breve{S}}^{L}\le\varphi_{\breve{S}}^{U}\le1 \tag{10}$$

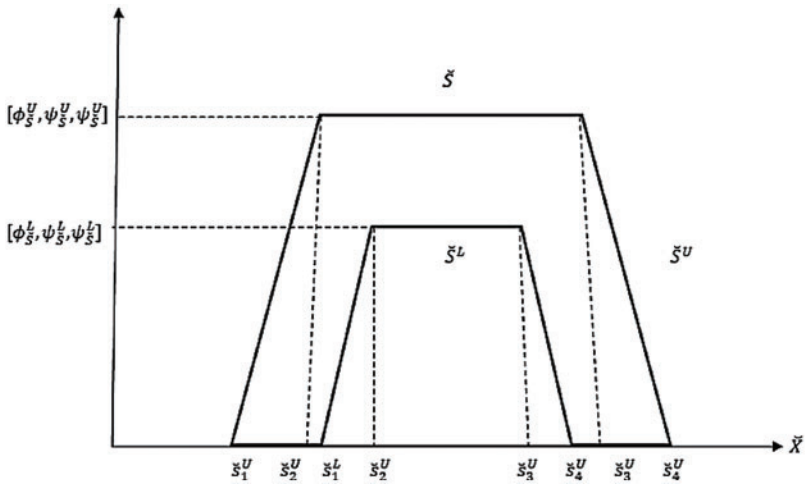

Figure 7: An interval type-2 trapezoidal neutrosophic number [19]

3.5 Step 5: IF_THEN Rule Building

A total of 27 rules are constructed using three layers of linguistic variables for the *MP*, *VF*, and *S* attributes of all transactions. Experts' expertise and knowledge are included in the formulation of the IF-THEN rules based on neutrosophic logic. Table 2 contains all of the rules. The rule that is fired is discovered after acquiring the membership degree of each component of the antecedent. The AND operator is used to get a single value when more than one portion of an antecedent is encountered for a particular rule. See [17,46] for more details.

Table 2: List of rules used in the neutrosophic logic-based deadlock resolution approach

Rule	Antecedent *MP*	Antecedent *VF*	Antecedent *S*	Consequent *TS*
1	High	High	High	Execute
2	High	High	Medium	Execute
3	High	High	Low	Execute
4	High	Medium	High	Execute
5	High	Medium	Medium	Execute
6	High	Medium	Low	Execute
7	High	Low	High	Execute
8	High	Low	Medium	Execute
9	High	Low	Low	Wait
10	Medium	High	High	Execute
11	Medium	High	Medium	Execute
12	Medium	High	Low	Execute
13	Medium	Medium	High	Execute
14	Medium	Medium	Medium	Wait
15	Medium	Medium	Low	Wait
16	Medium	Low	High	Execute
17	Medium	Low	Medium	Wait
18	Medium	Low	Low	Wait
19	Low	High	High	Execute
20	Low	High	Medium	Execute
21	Low	High	Low	Wait
22	Low	Medium	High	Execute
23	Low	Medium	Medium	Wait
24	Low	Medium	Low	Wait
25	Low	Low	High	Wait
26	Low	Low	Medium	Wait
27	Low	Low	Low	Wait

3.6 Step 6: Type-2 De-Neutrosophication

Prior to deneutrosophication, the range of output values is normalized to ensure that they are proportionally allocated to the neighboring neutrosophic transaction status (*TS*) sets [21]. Herein, a type-2 ordered weighted averaging (OWA) operator is utilized for the de-neutrosophication procedure [46]. An OWA operator having dimension n is a mapping $\breve{S}: \mathbb{R}^n \to \mathbb{R}$ associated with an n vector $\breve{W} = \left(\breve{W}_1, \breve{W}_2, \ldots, \breve{W}_n\right)$ such that $\breve{w}_i \in [0,1]$ and $\sum_{i=1}^{n} \breve{w}_i = 1$. Moreover,

$$\breve{S}_{\breve{w}}\left(\breve{S}_1, \breve{S}_2, \ldots, \breve{S}_n\right) = \sum_{j=1}^{n} \breve{w}_j \breve{t}_j \tag{11}$$

where $\breve{t}_j$ represents j-th largest element of aggregated object collection $\breve{S}_1, \breve{S}_2, \ldots, \breve{S}_n$. The (α, β, γ)-cut of an IT2NN $\breve{S}$ is represented as follows (see Fig. 8):

$$\breve{S} = \left\{ \langle \xi, \left(\left[\varnothing_{\breve{S}}^{U}(\xi), \varnothing_{\breve{S}}^{L}(\xi) \right] \geq \alpha, \left[\psi_{\breve{S}}^{U}(\xi), \psi_{\breve{S}}^{L}(\xi) \right] \leq \beta, \left[\varphi_{\breve{S}}^{U}(\xi), \varphi_{\breve{S}}^{L}(\xi) \right] \leq \gamma \right) \rangle \right\} \tag{12}$$

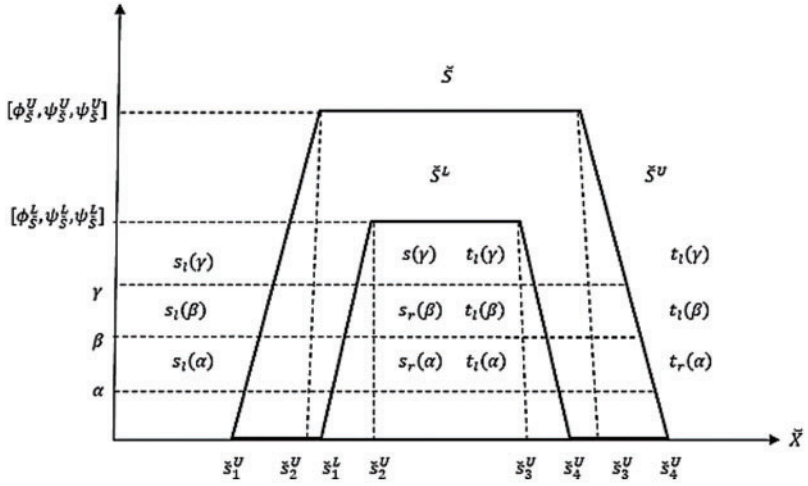

Figure 8: The $(\alpha; \beta; \gamma)$-cut of an IT2TrNN [19]

Fig. 9 summarizes the main architecture of the suggested deadlock model that makes use of transactional properties and neutrosophic logic to deal with the ambiguity of these characteristics to resolve deadlock.

4 Simulation Results

To assess the performance of the suggested deadlock resolution model, we performed a comprehensive series of simulation experiments using the matrix laboratory (MATLAB) and hypertext preprocessor (PHP) programming languages. The experiments were conducted on an x64-based processor and 8 Giga byte (GB) of double data rate 3 (DDR3) memory on an Intel ® CoreTM i7-5500 M CPU running at 2.50 GHz. A distributed real-time transaction processing simulator (DRTTPS) was used to conduct the simulations [47]. DRTTPS may be customized to generate a range of system loads and scenarios. An event is any action, such as granting locks, sending messages, or committing or aborting transactions. The simulated distributed system is made up of nodes that communicate with each other through messages on a virtual network. Pipes are used to connect nodes in this network. Pipes have a latency (to mimic transmission delay) and a bandwidth sufficient to transmit a certain number of messages. Each node has its own set of pages from which any transaction can read and/or write. The network's data is partitioned across nodes. At each node, a generator produces transactions at Poisson-like intervals. A more detailed description of DRTTPS's architecture can be found in [48].

The original DRTTPS will perform two simulations, one with an agent-based deadlock detection mechanism and another with a timeout-based deadlock detection strategy. The simulation has been updated to include the codes necessary to implement the suggested neutrosophic-based deadlock resolution paradigm. The results will be stored in a locally accessible my structured query language (MySQL) database. Configuration of the testbed is accomplished using an interactive user interface. Numerous load scenarios and system configurations can be simulated using a combination of

parameters. Additionally, the interface provides options for concurrency management, preemption, load balancing, and deadlock detection and resolution protocols. The simulation settings were used in accordance with [12]. While the suggested methodology is applicable to any real-time distributed database, we tested it on one in the area of health informatics. The data source is the Pima Indian diabetes dataset (PIDD) [49].

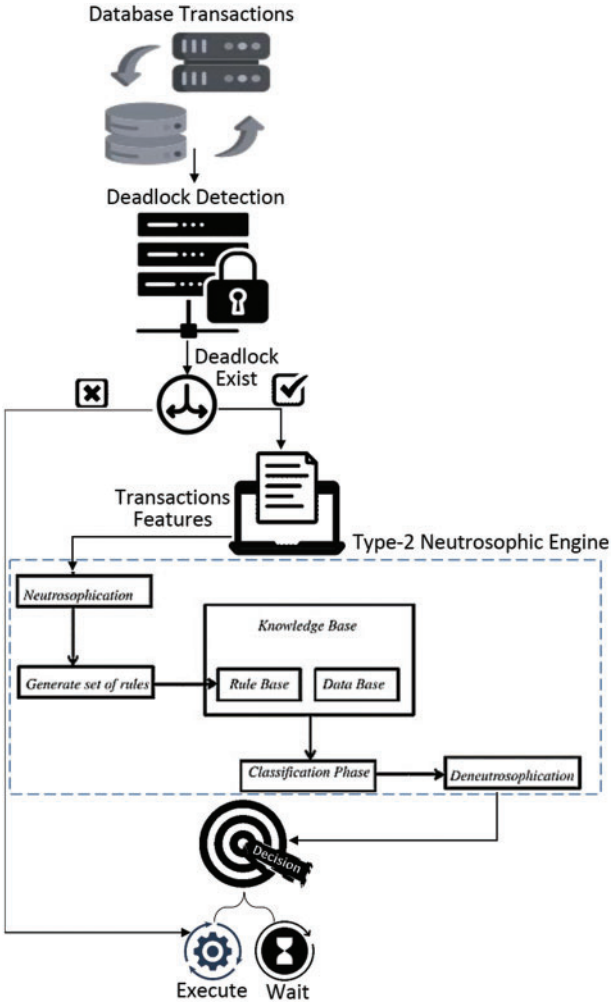

Figure 9: The proposed deadlock handling framework based on type-2 neutrosophic set

The first set of experiments was conducted to compare the suggested type-2 neutrosophic–based, type-1 neutrosophic-based, and traditional fuzzy-based deadlock resolution protocols in terms of execution rate. Herein, the simulator was running with a maximum of 25 transactions being allowed to be active at any given time. It was shown from Fig. 10 that the performance of type-2 neutrosophic-based resolution is better than the type-1 neutrosophic-based resolution on the execution ratio scale. The improvement rate has reached 10% to 20%, depending on the number of arrived transactions. It is also clear that the performance of the type 1-neutrosophic based approach outperforms the traditional fuzzy-based one. So, our resolution manager achieves up to 30x better throughput than the fuzzy resolution algorithm. One explanation for this performance is that both neutrosophic and

fuzzy based resolution approaches rely on transactions' features instead of serialization concepts. With six membership grades, type-2 neutrosophic logic effectively captures the ambiguity and produces solutions that are close to reality. It is feasible to make fewer wait decisions by incorporating a type-2 neutrosophic set in the model.

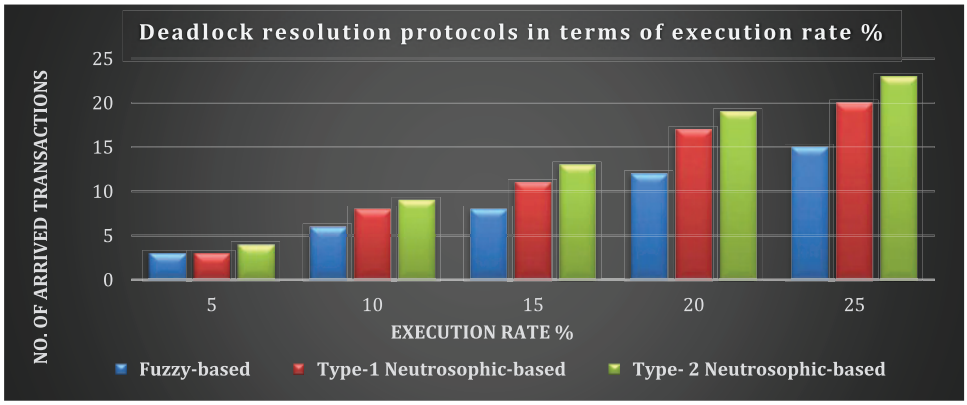

Figure 10: Comparison between fuzzy, type-1 neutrosophic, and type-2 neutrosophic–based deadlock resolution protocols in terms of execution rate %

The second set of experiments was run to evaluate the performance of the proposed type-2 neutrosophic-based deadlock resolution model with respect to the number of deadline-missed transactions in proportion to the total number of active transactions, according to the simulation outcomes. The results are listed in Table 3 in terms of executing and waiting rates, taking into account that the simulator was running with a maximum of 25 transactions being allowed to be active at any given time. The results reveal that the smaller the proportion of transactions that miss their deadline, the greater the chance of waiting for the transaction. When the number of transactions that miss their deadline increases, the proposed model attempts to reduce the number of waited transactions by merging the properties of the transactions, making a final decision on the transaction's overall priority, and scheduling the execution according to the priority vector of active transactions.

Table 3: Commit and wait rates for different transaction attributes

Total No. of active transactions/No. of deadline-miss transactions	Degree of deadline-miss transaction		Validation factor degree		Slackness degree	
	Wait	Execute	Wait	Execute	Wait	Execute
5/1	0	5	1	4	0	5
10/2	0	10	1	9	0	10
15/4	3	12	1	14	1	14
20/4	2	18	2	18	2	18
25/5	3	22	3	22	2	23

In the same scenario, the results in Table 3 reveal the performance of the proposed model's in terms of executing and waiting rates with respect to the other two degrees: validation factor degree and slackness degree, in proportion to the total number of active transactions, according to the simulation outcomes. As expected, the results confirm that the lower the validation factor degree, the longer the deadline is, the probability of waiting for the transaction increases. What's more, the results indicate that the more slackness a transaction has, the higher its priority should be. In this situation, the earlier the transaction's deadline, the greater its priority.

5 Conclusion

The suggested approach addresses the shortcomings of previous systems by prioritizing transactions according to their characteristics. The model makes use of the concepts of validation factor degree, slackness degree, and degree of deadline-missed transaction to improve real-time performance and prioritize transactions with a higher data conflict resolution importance. Type-2 neutrosophic logic is used to combine the attributes of transactions in order to facilitate conflict resolution between them. The amount of uncertainty that occurs in the value of a grade of membership is represented here by the footprint of uncertainty for truth, indeterminacy, and falsity. The simulation implementation on a medical PIMA Indian diabetes database and a performance comparison between type-1 and type 2 neutrosophic-based real-time deadlock control methods show that our method can ensure good real-time performance while guaranteeing temporal consistency. In future work, we plan to develop a hybrid framework between a neutrosophic and a rule-based system [50,51] that incorporates deep learning. Furthermore, more experiments will be conducted to test the efficiency of the proposed model with other membership functions or a higher number of membership functions.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] E. Mourtou, "An evaluation of deadlock situations in a Greek hospital database," *Journal on Information Technology in Healthcare*, vol. 7, no. 6, pp. 400–410, 2009.

[2] W. Gardner, K. Fierlbeck and A. Levy, "Breaking the deadlock: Towards a new intergovernmental relationship in Canadian healthcare," *Healthc Pap*, vol. 14, pp. 7–15, 2014.

[3] R. Jørgensen, J. Christiansen, H. Nissen, K. Kristoffersen and V. Zoffmann, "The deadlock of saying "that is what we already do!" A thematic analysis of mental healthcare professionals' reactions to using an evidence-based intervention," *Journal of Psychiatric and Mental Health Nursing*, vol. 26, no. 1–2, pp. 39–48, 2019.

[4] D. Jasmina, Z. Avdagic, F. Orucevic and S. Omanovic, "Advanced consistency management of highly-distributed transactional database in a hybrid cloud environment using novel R-TBC/RTA approach," *Journal of Cloud Computing*, vol. 10, no. 1, pp. 1–31, 2021.

[5] D. Jörg, C. Hauser and B. Erb, "Reliability and availability properties of distributed database systems," in *Proc. of the 18th Int. Enterprise Distributed Object Computing Conf.*, USA, pp. 226–233, 2014.

[6] T. Sashi, R. Batth and S. Kaur, "A review on fragmentation, allocation and replication in distributed database systems," in *Proc. of the Int. Conf. on Computational Intelligence and Knowledge Economy*, UAE, pp. 538–544, 2019.

[7] P. Shanchen, H. Chen, H. Liu, J. Yao and M. Wang, "A deadlock resolution strategy based on spiking neural P systems," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 1, pp. 1–12, 2019.
[8] D. Ola, "Analysis of deadlock detection and resolution algorithms in distributed database system," *African Journal of Computing & ICT*, vol. 8, no. 1, pp. 205–211, 2015.
[9] E. Janani and M. Khalili, "A new logistic algorithm contrasts transactions deadlock by using mamdani controller," in *Proc. of the Int. Conf. on Knowledge-Based Engineering and Innovation*, Iran, pp. 1–10, 2017.
[10] J. Sumika, N. Kumar and K. Chauhan, "An overview on deadlock resolution techniques," *International Journal of Engineering Research & Technology*, vol. 7, pp. 1–4, 2019.
[11] M. Deepti, "Deadlock prevention algorithm in grid environment," in *Proc. of MATEC Web of Conferences*, India, pp. 20113–20119, 2016.
[12] H. Waqar, A. Vezina and M. Fontaine, "Topological response to deadlock detection and resolution in real-time database systems," in *Proc. of the IEEE Int. Conf. on Internet of Things*, Egypt, pp. 1880–1887, 2018.
[13] G. Masoomeh and A. Harounabadi, "A new method for optimization of deadlock resolution of distributed database with formal model," *International Journal of Electronics Communication and Computer Engineering*, vol. 5, no. 1, pp. 220–228, 2014.
[14] S. Darwish, A. El-Zoghabi and M. Hassan, "Soft computing for database deadlock resolution," *International Journal of Modeling and Optimization*, vol. 5, no. 1, pp. 15–21, 2015.
[15] M. Abdel-Basset, M. Mohamed, F. Smarandache and V. Chang, "Neutrosophic association rule mining algorithm for big data analysis," *Symmetry*, vol. 10, no. 4, pp. 1–19, 2018.
[16] S. De and J. Mishra, "Processing of inconsistent neutrosophic data and selecting primary key from the relation," in *Proc. of IEEE Int. Conf. on Inventive Computing and Informatics*, India, pp. 317–320, 2017.
[17] D. Nagarajan, M. Lathamaheswari, S. Broumi and J. Kavikumar, "A new perspective on traffic control management using triangular interval type-2 fuzzy sets and interval neutrosophic sets," *Operations Research Perspectives*, vol. 6, no. 100099, pp. 1–13, 2019.
[18] F. Karaaslan and F. Hunu, "Type-2 single-valued neutrosophic sets and their applications in multi-criteria group decision making based on TOPSIS method," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 10, pp. 4113–4132, 2020.
[19] M. Touqeer, R. Umer, A. Ahmadian and S. Salahshour, "A novel extension of TOPSIS with interval type-2 trapezoidal neutrosophic numbers using (α, β, γ)-cuts," *RAIRO-Operations Research*, vol. 55, no. 5, pp. 2657–2683, 2021.
[20] A. Bakali, S. Broumi, D. Nagarajan, M. Talea, M. Lathamaheswari *et al.,* "Graphical representation of type-2 neutrosophic sets," *Neutrosophic Sets and Systems*, vol. 42, pp. 28–38, 2021.
[21] S. Pai and R. Gaonkar, "Safety modelling of marine systems using neutrosophic logic," *Journal of Engineering for the Maritime Environment*, vol. 235, pp. 225–235, 2021.
[22] S. Topal, F. Tas, S. Broumi and O. Kirecci, "Applications of neutrosophic logic of smart agriculture via internet of things," *International Journal of Neutrosophic Science*, vol. 12, no. 2, pp. 105–115, 2020.
[23] K. Al-Hussaini, N. Al-Amdi and F. Abdulrazzak, "A new multi-resource deadlock detection algorithm using directed graph requests in distributed database systems," in *Proc. of the Int. Conf. of Reliable Information and Communication Technology*, Malaysia, pp. 462–474, 2020.
[24] S. Gupta, "Deadlock detection techniques in distributed database system," *International Journal of Computer Applications*, vol. 74, no. 21, pp. 41–44, 2013.
[25] A. Rashid and N. Ali, "Deadlock detection and resolution in distributed database environment," *International Journal of Scientific and Research Publications*, vol. 5, no. 9, pp. 1–9, 2015.
[26] P. Chahar and S. Dalal, "Deadlock resolution techniques: An overview," *International Journal of Scientific and Research Publications*, vol. 3, no. 7, pp. 1–5, 2013.
[27] J. Nithiya and S. Priya, "Thread based deadlock detection and management in distributed database," *Journal of Engineering Research and Application*, vol. 8, no. 2, pp. 56–61, 2018.

[28] S. Pashazadeh, "Modeling and verification of deadlock potentials of a concurrency control mechanism in distributed databases using hierarchical colored petri net," *International Journal of Information and Education Technology*, vol. 2, no. 2, pp. 77–82, 2012.
[29] M. Goswami, K. Vaisla and A. Singh, "VGS algorithm: An efficient deadlock prevention mechanism for distributed transactions using pipeline method," *International Journal of Computer Applications*, vol. 46, no. 22, pp. 1–9, 2012.
[30] A. Ansari, R. Biswas and S. Aggarwal, "Neutrosophic classifier: An extension of fuzzy classifier," *Applied Soft Computing*, vol. 13, no. 1, pp. 563–573, 2013.
[31] B. Kavitha, S. Karthikeyan and P. Maybell, "An ensemble design of intrusion detection system for handling uncertainty using neutrosophic logic classifier," *Knowledge-Based Systems*, vol. 28, no. 1, pp. 88–96, 2012.
[32] S. Saravanakumar, "A real time approach on genetically evolving intrusion detection using neutrosophic logic inference system," in *Proc. of the Int. Conf. on Computing Intelligence and Data Science*, India, pp. 49–62, 2018.
[33] A. Hefny, A. Hassanien and S. Basha, "Neutrosophic rule-based identity verification system based on handwritten dynamic signature analysis," *Computers, Materials & Continua*, vol. 69, no. 2, pp. 2367–2385, 2021.
[34] M. Abdel-Basset, M. Gunasekaran, M. Mohamed and F. Smarandache, "A novel method for solving the fully neutrosophic linear programming problems," *Neural Computing and Applications*, vol. 31, no. 5, pp. 1595–1605, 2019.
[35] S. Kalita, M. Kalita and S. Sarmah, "A survey on distributed deadlock detection algorithm and its performance evolution," *International Journal of Innovative Science, Engineering & Technology*, vol. 2, no. 4, pp. 615–620, 2015.
[36] M. Grechanik, B. Hossain and H. Wang, "Preventing database deadlocks in applications," in *Proc. of the 9th Joint Meeting on Foundations of Software Engineering*, Russia, pp. 356–366, 2013.
[37] W. Lu, C. Yu, W. Xing, X. Che and Y. Yang, "An efficient deadlock detection and resolution algorithm for generalized deadlocks," *International Journal of Innovative Computing, Information and Control*, vol. 13, no. 2, pp. 703–710, 2017.
[38] W. Lu, Y. Yang, L. Wang, W. Xing, X. Che *et al.,* "A fault tolerant election-based deadlock detection algorithm in distributed systems," *Software Quality Journal*, vol. 26, no. 3, pp. 991–1013, 2017.
[39] X. Zhang and M. Uzam, "Transition-based deadlock control policy using reachability graph for flexible manufacturing systems," *Advances in Mechanical Engineering*, vol. 8, no. 2, pp. 1–9, 2016.
[40] S. Pandey and U. Shanker, "Transaction scheduling protocols for controlling priority inversion: A review," *Computer Science Review*, vol. 35, pp. 1–15, 2020.
[41] N. Krivokapić, A. Kemper and E. Gudes, "Deadlock detection in distributed database systems: A new algorithm and a comparative performance analysis," *The VLDB Journal*, vol. 8, no. 2, pp. 79–100. 1999.
[42] M. Haroon, "Challenges of concurrency control in object oriented distributed database systems," *International Journal of Modern Computation, Information and Communication Technology*, vol. 2, no. 7, pp. 48–52, 2019.
[43] S. Pandey and U. Shanker, "RACE: A concurrency control protocol for time-constrained transactions," *Arabian Journal for Science and Engineering*, vol. 45, pp. 10131–10146, 2020.
[44] W. Moudani, N. Khoury and M. Hussein, "An optimistic concurrency control approach applied to temporal data in real-time database systems," *WSEAS Transactions on Computers*, vol. 11, no. 12, pp. 419–434, 2012.
[45] M. Khatib and M. Atique, "FGSA for optimal quality of service-based transaction in real-time database systems under different workload condition," *Cluster Computing*, vol. 23, no. 1, pp. 307–319, 2020.
[46] P. Singh, "A type-2 neutrosophic-entropy-fusion based multiple thresholding method for the brain tumor tissue structures segmentation," *Applied Soft Computing*, vol. 103, no. 107119, pp. 1–23, 2021.
[47] M. Samani, "*Distributed Database System-Simulator*," California, USA: GitHub, Inc., 2022. [Online]. Available: https://github.com/mani-samani/DistributedDatabaseSystem-Simulator.

[48] W. Haque and P. Stokes, "Simulation of a complex distributed real-time database system," in *Proc. of the Spring Simulation Multi-Conf.*, UK, pp. 359–366, 2007.
[49] D. Aha, "*UC Irvine Machine Learning Repository*," California, USA: National Science Foundation, University of California, 2007. [Online]. Available: https://archive.ics.uci.edu/ml/datasets.php.
[50] H. Sun and R. Grishman, "Lexicalized dependency paths based supervised learning for relation extraction," *Computer Systems Science and Engineering*, vol. 43, no. 3, pp. 861–870, 2022.
[51] H. Sun and R. Grishman, "Employing lexicalized dependency paths for active learning of relation extraction," *Intelligent Automation & Soft Computing*, vol. 34, no. 3, pp. 1415–1423, 2022.